

# Technical Specifications Document

## **“AutoBuild” - Automatic PC Part Builder**

October 10, 2020

### Saturday Solution

Connor Kobel	(015036474)
Daniel Gulland	(018648771)
Ian Ho-Sing-Loy	(026339220)
Nickolaus Marshall-Eminger	(025567241)
Sirage El-Jawhari	(018359144)
Zeinab Farhat	(017990360)

## Table of Contents

<b>1 Document Revisions</b>	<b>4</b>
<b>2 Saturday Solutions Member Directory</b>	<b>5</b>
<b>3 General Technologies</b>	<b>6</b>
Integrated development environment	6
What We Use	6
Why VS 2019?	6
Get VS 2019 CE for Windows	7
Change the installer language from the command line	10
Get VS 2019 for macOS	13
Language	16
What We Use	16
What is C#?	16
Why C#?	16
Operating System	17
What We Use	17
Why?	17
How will we handle updates?	17
<b>4 Specific Technologies</b>	<b>18</b>
Database Client	19
SQL Server Management Studio	19
Why use SMSS?	19
Version information	19
Get SQL Server Management Studio	19
Download SQL Server Management Studio (SSMS)	19
Azure Data Studio	19
Why use Azure?	19
Version Information	20
Supported SQL offerings	20
Get Azure Data Studio for Windows	20
Get Azure Data Studio for macOS	20
Get Azure Data Studio for Linux	21
Which to use?	22
How will we handle updates?	22
How will we fix it?	22
Database Engine	23
SQL Server 2019	23
What we use	23

Installation	23
Why use SQL Server 2019 Developer Edition?	24
How will we handle updates?	24
How will we fix it?	24
Framework	25
.NET Core 3.1	25
Why Use NET Core 3.1?	25
Install on Windows	25
Install on MacOS	25
Install on Linux	25
What happens with an update?	25
How do we fix it?	25
Test Browser	26
Google Chrome	26
What is Chrome?	26
Why use Chrome?	26
How to Install Google Chrome	26
What are DevTools?	27
Chrome DevTools	27
How to get DevTools	27
How to use DevTools	27
Do I want to use DevTools Extension?	28
What happens with an update?	28
How do we fix it?	28
Web Server	30
What is ASP.Net Cores Kestrel	30
<b>5 Appendices</b>	<b>44</b>

# **1 Document Revisions**

<b>Date</b>	<b>Version</b>	<b>Changes</b>
10/7/2020	0.1.0	Initial Draft formatting.
10/8/2020	0.2.0	Added rough specifications for each element.
10/9/2020	0.3.0	Added more detailed specs, cleaned up current entries.
10/10/2020	1.0.0	All data added and formatted.

## **2 Saturday Solutions Member Directory**

<b>Name</b>	<b>Student ID</b>	<b>Email</b>
Connor Kobel	015036474	crkobel@verizon.net
Daniel Gulland	018648771	dannygulland@gmail.com
Ian Ho-Sing-Loy	026339220	ian.hosingloy@gmial.com
Nick Marshall-Eminger	025567241	hunter3787@gmail.com
Sirage El-Jwhari	018359144	sergejawhari@gmail.com
Zeinab Farhat	017990360	Zeinab.farhat@student.csulb.edu

## **3 General Technologies**

### **Integrated development environment**

#### **What We Use**

We will be using Visual Studio 2019 Community Edition to develop with.

Version:

- Windows: 16.7.5
- Mac: 8.7.8 (build 4)

#### **Why VS 2019?**

Visual Studio 2019 is free to download and use, is fairly powerful, and has built in tools to help with SQL management. VS2019 supports the C# language, which we intend to use for our product, as well as carrying many productivity features.

Visual Studio 2019 version 16.7 is the third supported servicing baseline for Visual Studio 2019. Enterprise and Professional customers needing to adopt a long term stable and secure development environment are encouraged to standardize on this version. As explained in more detail in our [lifecycle and support policy](#), version 16.7 will be supported with fixes and security updates for one year after the release of the next servicing baseline.

In addition, now that version 16.7 is available, version 16.4, which was the last servicing baseline, will be supported for an additional 12 months and will go out of support in October 2021. Note as well that versions 16.5 and 16.6 are no longer under support. These intermediary releases received servicing fixes only until the next minor update was released.

You can acquire the latest most secure version of Visual Studio 2019 version 16.7 in the [downloads section of my.visualstudio.com](#). For more information about Visual Studio supported baselines, please review the [support policy for Visual Studio 2019](#).

## Get VS 2019 CE for Windows

### Step 1 - Make sure your computer is ready for Visual Studio

Before you begin installing Visual Studio:

1. Check the [system requirements](#). These requirements help you know whether your computer supports Visual Studio 2019.
2. Apply the latest Windows updates. These updates ensure that your computer has both the latest security updates and the required system components for Visual Studio.
3. Reboot. The reboot ensures that any pending installs or updates don't hinder the Visual Studio install.
4. Free up space. Remove unneeded files and applications from your %SystemDrive% by, for example, running the Disk Cleanup app.

For questions about running previous versions of Visual Studio side by side with Visual Studio 2019, see the [Visual Studio 2019 Platform Targeting and Compatibility](#) page.

### Step 2 - Download Visual Studio

Next, download the Visual Studio bootstrapper file.

To do so, choose the following button, choose the edition of Visual Studio that you want, choose Save, and then choose Open folder.

[Download Visual Studio](#)

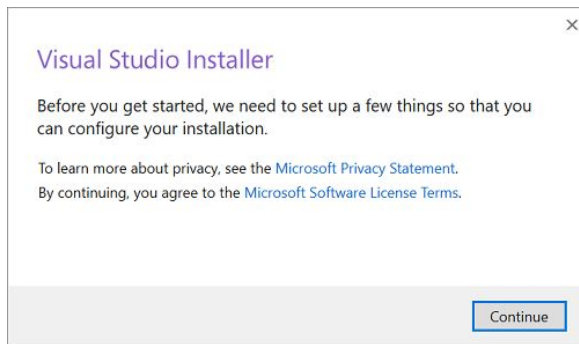
Above links web address: <https://visualstudio.microsoft.com/downloads>

### Step 3 - Install the Visual Studio installer

Run the bootstrapper file to install the Visual Studio Installer. This new lightweight installer includes everything you need to both install and customize Visual Studio.

1. From your Downloads folder, double-click the bootstrapper that matches or is similar to one of the following files:
  - vs\_community.exe for Visual Studio Community
  - vs\_professional.exe for Visual Studio Professional
  - vs\_enterprise.exe for Visual Studio Enterprise
2. If you receive a User Account Control notice, choose Yes.

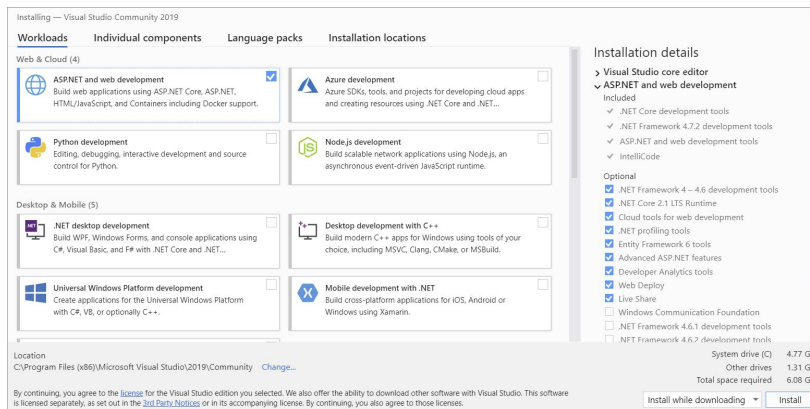
3. We'll ask you to acknowledge the Microsoft [License Terms](#) and the Microsoft [Privacy Statement](#). Choose Continue.



## Step 4 - Choose workloads

After the installer is installed, you can use it to customize your installation by selecting the feature sets—or workloads—that you want. Here's how.

1. Find the workload you want in the Visual Studio Installer.



For example, choose the "ASP.NET and web development" workload. It comes with the default core editor, which includes basic code editing support for over 20 languages, the ability to open and edit code from any folder without requiring a project, and integrated source code control.

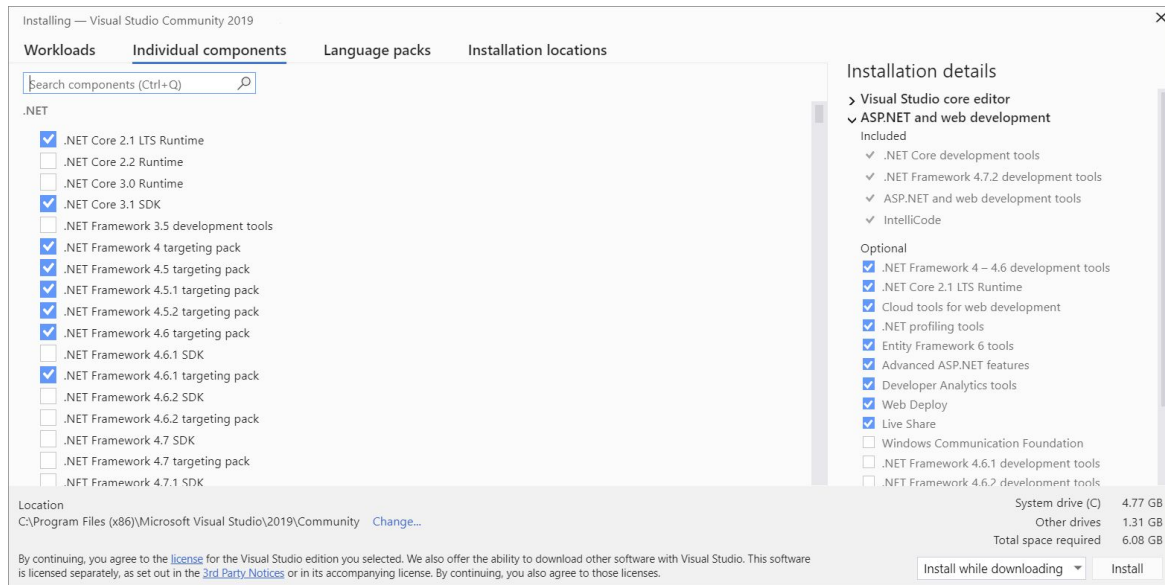
2. After you choose the workload(s) you want, choose Install.  
Next, status screens appear that show the progress of your Visual Studio installation.

**Tip:** At any time after installation, you can install workloads or components that you didn't install initially. If you have Visual Studio open, go to Tools > Get Tools and Features... which opens the Visual Studio Installer. Or, open Visual Studio Installer from the Start menu. From there, you can choose the workloads or components that you wish to install. Then, choose Modify.



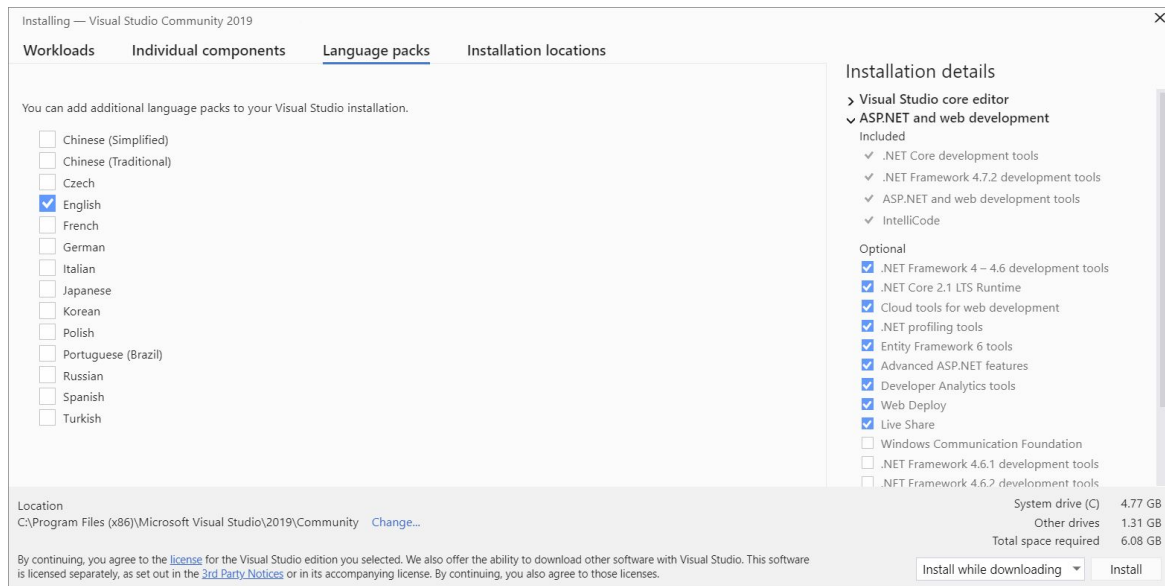
## Step 5 - Choose individual components (Optional)

If you don't want to use the Workloads feature to customize your Visual Studio installation, or you want to add more components than a workload installs, you can do so by installing or adding individual components from the Individual components tab. Choose what you want, and then follow the prompts.



## Step 6 - Install language packs (Optional)

By default, the installer program tries to match the language of the operating system when it runs for the first time. To install Visual Studio in a language of your choosing, choose the Language packs tab from the Visual Studio Installer, and then follow the prompts.

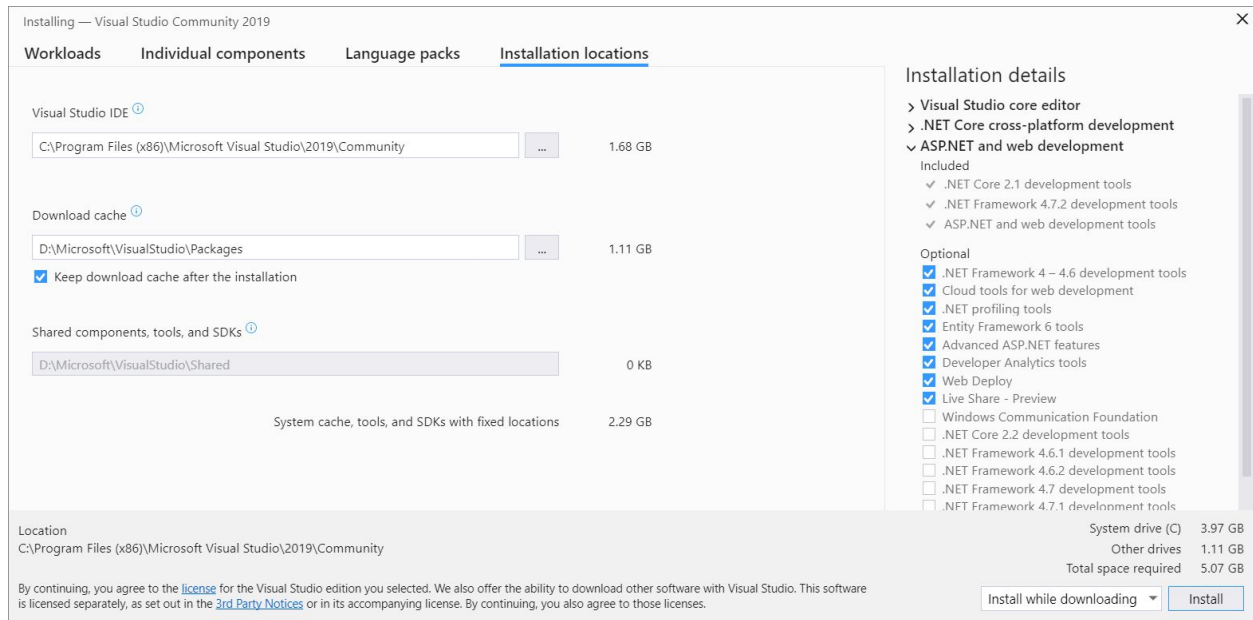


## Change the installer language from the command line

Another way that you can change the default language is by running the installer from the command line. For example, you can force the installer to run in English by using the following command: `vs_installer.exe --locale en-US`. The installer will remember this setting when it is run the next time. The installer supports the following language tokens: zh-cn, zh-tw, cs-cz, en-us, es-es, fr-fr, de-de, it-it, ja-jp, ko-kr, pl-pl, pt-br, ru-ru, and tr-tr.

## Step 7 - Select the installation location (Optional)

You can reduce the installation footprint of Visual Studio on your system drive. You can choose to move the download cache, shared components, SDKs, and tools to different drives, and keep Visual Studio on the drive that runs it the fastest.



**Important:** You can select a different drive only when you first install Visual Studio. If you've already installed it and want to change drives, you must uninstall Visual Studio and then reinstall it.

For more information, see the [Select installation locations](#) page.

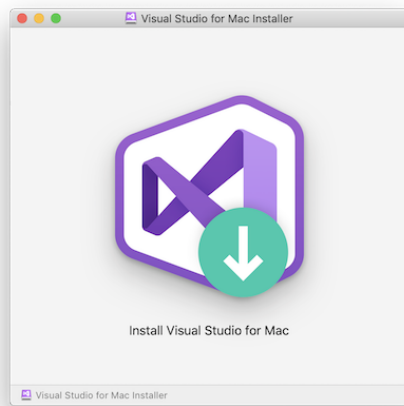
## **Step 8 - Start developing**

1. After Visual Studio installation is complete, choose the Launch button to get started developing with Visual Studio.
2. On the start window, choose Create a new project.
3. In the search box, enter the type of app you want to create to see a list of available templates. The list of templates depends on the workload(s) that you chose during installation. To see different templates, choose different workloads. You can also filter your search for a specific programming language by using the Language drop-down list. You can filter by using the Platform list and the Project type list, too.
4. Visual Studio opens your new project, and you're ready to code!

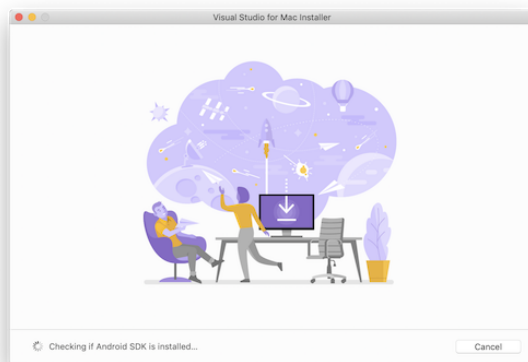
## Get VS 2019 for macOS

### Installation instructions

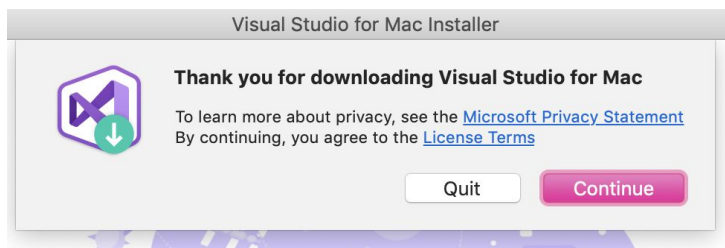
1. Download the installer from the [Visual Studio for Mac download page](#).
2. Once the download is complete, click the VisualStudioforMacInstaller.dmg to mount the installer, then run it by double-clicking the arrow logo:



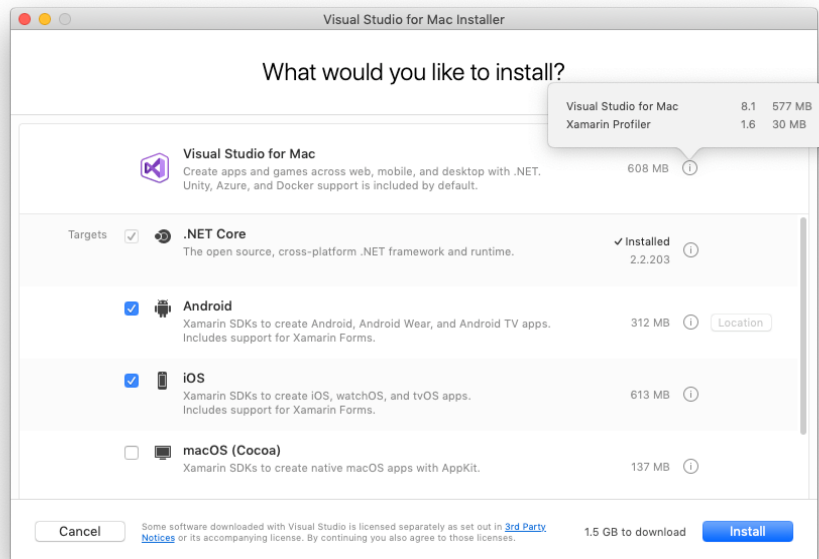
3. You may be presented with a warning about the application being downloaded from the Internet. Click Open.
4. Wait while the installer checks your system:



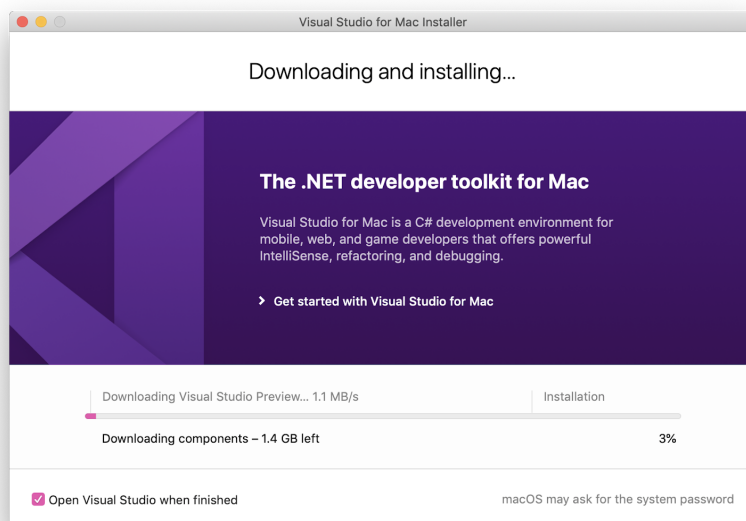
5. An alert will appear asking you to acknowledge the privacy and license terms. Follow the links to read them, then press Continue if you agree:



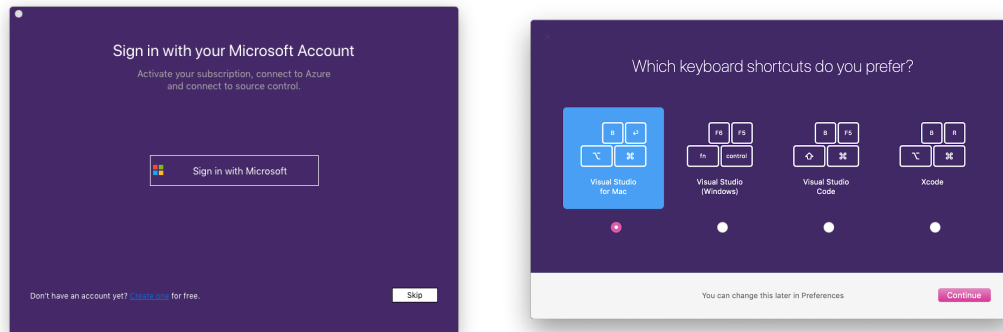
- The list of available workloads is displayed. Select the components you wish to use:



- After you have made your selections, press the Install button.
- The installer will display progress as it downloads and installs Visual Studio for Mac and the selected workloads. You will be prompted to enter your password to grant the privileges necessary for installation.:



9. Once installed, Visual Studio for Mac will prompt you to personalize your installation by signing in and selecting the key bindings that you'd like to use:



If you have network trouble while installing in a corporate environment, review the [installing behind a firewall or proxy](#) instructions.

Learn more about the changes in the [release notes](#).

# Language

## What We Use

We will be programming primarily in C#.

## What is C#?

C# is a new language created by Microsoft and submitted to the ECMA for standardization. This new language was created by a team of people at Microsoft led by Anders Hejlsberg . Interestingly, Hejlsberg is a Microsoft Distinguished Engineer who has created other products and languages, including Borland Turbo C++ and Borland Delphi. With C#, they focused on taking what was right about existing languages and adding improvements to make something better.

C# is a powerful and flexible programming language. Like all programming languages, it can be used to create a variety of applications. Your potential with C# is limited only by your imagination. The language does not place constraints on what you can do. C# has already been used for projects as diverse as dynamic Web sites, development tools, and even compilers.

C# was created as an object-oriented programming (OOP) language. Other programming languages include object-oriented features, but very few are fully object-oriented. In my book you can learn how C# compares to some of these other programming languages. My book also covers what it means to use an object-oriented language as well as the details of doing object-oriented programming with C#

(Taken from: <https://www.developer.com/net/asp/article.php/922211/what-is-c.htm>)

## Why C#?

C# is an object-oriented language, similar to java and C++, but it handles much of the overhead that must be handled in C++ programming, yet it's a compiled language, unlike java. This means that before the application launches on the server the code must be converted to binary. This means the application should operate faster after being compiled and run.

C# is used for a wide array of applications, but one of the top uses is server-side web applications, which is what we intend to create here. ASP.NET is a framework leveraged in this product that uses the C# language, making it a logical and comfortable choice for us to program with.



## Operating System

### What We Use

We will be using Windows and MacOS for development.

Operating System Version	Build Number
Windows Home: 1903	18362.1082
Windows Pro: 2004	19041.508
MacOS: "Catalina"	10.15.7

### Why?

We are using Windows since a majority of our personal computers are already Windows PCs, and will be time and cost effective to work with what we have. Most of our software will work on either platform, since Visual Studio has a working mac version with most of the same functionality.

### How will we handle updates?

In the case of system updates we will be holding off on applying them until we can verify they are not product breaking. Generally we hope this will not be a problem, but it is widely known that with a big macOS upgrade some programs just will not function at all. In the case of Windows updates we will go through with any security and ensure our systems do not become insecure, but larger build updates will be held until performance can be assured.

## **4 Specific Technologies**

- I. Database Client**
- II. Database Engine**
- III. Framework**
- IV. Test Browser**
- v. Web Server**

## **Database Client**

### **SQL Server Management Studio**

SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. Use SSMS to access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, and SQL Data Warehouse.

#### **Why use SMSS?**

SSMS provides a single comprehensive utility that combines a broad group of graphical tools with a number of rich script editors to provide access to SQL Server for developers and database administrators of all skill levels.

SMSS is cost-free, has various add-on options, and has an easy installation.

#### **Version information**

- Release number: 18.6
- Build number: 15.0.18338.0
- Release date: July 22, 2020

#### **Get SQL Server Management Studio**

Follow the link below and the steps provided:

[Download SQL Server Management Studio \(SSMS\)](#)

If the link does not work the web address below can be used:

<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>

### **Azure Data Studio**

Azure Data Studio is a cross-platform database tool for data professionals using the Microsoft family of on-premises and cloud data platforms on Windows, MacOS, and Linux.

#### **Why use Azure?**

Azure Data Studio offers a modern editor experience with IntelliSense, code snippets, source control integration, and an integrated terminal. It's engineered with the data platform user in mind, with built-in charting of query result sets and customizable dashboards.

The source code for Azure Data Studio and its data providers is available on GitHub under a source code EULA that provides rights to modify and use the software, but not to redistribute it or host it in a cloud service.

## Version Information

- Build number: 1.22.1
- Release date: September 30, 2020

Available for Windows, macOS, and Linux

## Supported SQL offerings

This version of Azure Data Studio works with all [supported versions of SQL Server 2014 - SQL Server 2019 \(15.x\)](#) and provides support for working with the latest cloud features in Azure SQL Database and Azure SQL Data Warehouse. Azure Data Studio also provides preview support for Azure SQL Managed Instance.

## Get Azure Data Studio for Windows

This release of Azure Data Studio includes a standard Windows Installer experience, and a .zip file.

The *user installer* is recommended because it does not require administrator privileges, which simplify both installs and upgrades. The user installer does not require Administrator privileges as the location is under your user Local AppData (LOCALAPPDATA) folder. The user installer also provides a smoother background update experience. For more information, see [User setup for Windows](#).

### User Installer (recommended)

1. Download and run the [Azure Data Studio user installer for Windows](#).
2. Start the Azure Data Studio app.

### System Installer

1. Download and run the [Azure Data Studio system installer for Windows](#).
2. Start the Azure Data Studio app.

### .zip file

1. Download [Azure Data Studio .zip for Windows](#).
2. Browse to the downloaded file and extract it.
3. Run `\azuredatstudio-windows\azuredatstudio.exe`

## Get Azure Data Studio for macOS

1. Download [Azure Data Studio for macOS](#).
2. To expand the contents of the zip, double-click it.
3. To make Azure Data Studio available in the *Launchpad*, drag *Azure Data Studio.app* to the *Applications* folder.

### Get Azure Data Studio for Linux

1. Download Azure Data Studio for Linux by using one of the installers or the tar.gz archive:
  - [.deb](#)
  - [.rpm](#)
  - [.tar.gz](#)
2. To extract the file and launch Azure Data Studio, open a new Terminal window and type the following commands:

#### Debian Installation:

- `cd ~`
- `sudo dpkg -i ./Downloads/azuredatstudio-linux-<version string>.deb`
- `azuredatstudio`

#### rpm Installation:

- `cd ~`
- `yum install ./Downloads/azuredatstudio-linux-<version string>.rpm`
- `azuredatstudio`

#### Tar.gz Installation:

- `cd ~`
- `cp ~/Downloads/azuredatstudio-linux-<version string>.tar.gz ~`
- `tar -xvf ~/azuredatstudio-linux-<version string>.tar.gz`
- `echo 'export PATH="$PATH:~/azuredatstudio-linux-x64"' >> ~/.bashrc`
- `source ~/.bashrc`
- `azuredatstudio`

### NOTE!

On Debian, Redhat, and Ubuntu, you may have missing dependencies. Use the following commands to install these dependencies depending on your version of Linux:

#### Debian:

```
sudo apt-get install libunwind8
```

#### Redhat:

```
yum install libXScrnSaver
```

**Ubuntu:**

- `sudo apt-get install libxss1`
- `sudo apt-get install libgconf-2-4`
- `sudo apt-get install libunwind8`

**Which to use?**

Use Azure Data Studio if you:

- Need to run on macOS or Linux
- Are connecting to a SQL Server 2019 big data cluster
- Spend most of your time editing or executing queries
- Need the ability to quickly chart and visualize result sets
- Can execute most administrative tasks via the integrated terminal using `sqlcmd` or Powershell
- Have minimal need for wizard experiences
- Don't need to do deep administrative configuration

Use SQL Server Management Studio if you:

- Spend most of your time on database administration tasks
- Are doing deep administrative configuration
- Are doing security management, including user management, vulnerability assessment, and configuration of security features
- Make use of the Reports for SQL Server Query Store
- Need to make use of performance tuning advisors and dashboards
- Are doing Import/Export of DACPACs
- Need access to Registered Servers and want to control SQL Server services on Windows

**How will we handle updates?**

At this time we plan to use some of the most recent builds and versions so as to avoid updating for quite some time, however in the case that updates do occur we will check the change log to determine if there are any fundamental updates that could break our system.

**How will we fix it?**

With the change log readily available we will migrate to accommodate the changes necessary. For each depreciated method/function/etc. we will have to rewrite to adjust for the modern methods.

# **Database Engine**

## **SQL Server 2019**

### **What we use**

We will be using SQL Server 2019 Developer edition.

### **Installation**

#### **For Mac users:**

- 1) Install Docker -  
<https://hub.docker.com/editions/community/docker-ce-desktop-mac?tab=description>
- 2) Open terminal and execute the following command -  
`sudo docker pull mcr.microsoft.com/mssql/server:2019-CTP2.2-ubuntu`
- 3) Also in terminal, execute the following command -  
`sudo docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=MyStrongPassword@1' -p 1433:1433 --name sqlserver2019 -d mcr.microsoft.com/mssql/server:2019-CTP2.2-ubuntu`

#### **Things to note:**

- i) -e ACCEPT\_EULA=Y indicates that you agree to Microsoft's EUA (End User Licence Agreement).
  - ii) -e SA\_PASSWORD is where you set the system administrator password for SQL Server. The password must be at least 8 characters long and contain characters from three of the following four sets: uppercase letters, lowercase letters, numbers & symbols.
  - iii) -p flag allows the 1433 to be used as the TCP port number.
  - iv) --name sets the instance name to sqlserver2019.
  - v) -d runs docker in daemon mode, used to run the container in the background.
- 4) Execute the following command -  
`sudo docker exec -it sqlserver2019 "bash"`
  - 5) Execute the following command -  
`/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P 'MyStrongPassword@1'`

#### **For Windows users:**

- 1) Install SQL Server Developer edition -  
<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>
- 2) Run the executable.
- 3) Select Basic installation and select download.

## **Why use SQL Server 2019 Developer Edition?**

SQL Server Developer edition lets developers build any kind of application on top of SQL Server. It includes all the functionality of the Enterprise edition but is licensed for use as a development and test system, not as a production server. SQL Server Developer is an ideal choice for people who build and test applications. SQL Server works natively with .NET applications, which makes integration simpler, so it's the choice for software that runs on a Windows server or desktop. For example, LINQ (Language Integrated Query) is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages. With SQL Server, you can easily set up your entity framework classes and get started, but with other RDBMS such as MySQL, you would need to download third party tools.

## **How will we handle updates?**

We plan on using the most recent version to avoid updating. SQL Server continues support until 2030. However, if updates occur, we will run all of our tests to ensure our application works as intended.

## **How will we fix it?**

In the case that our system breaks, we will update each piece of code to get the system back up and running.



## **Framework**

### **.NET Core 3.1**

We will be utilizing .NET Core 3.1

### **Why Use NET Core 3.1?**

.NET Core is designed to remove some of the limitations that come with .NET framework, such as only working on a windows desktop, or requiring specific APIs for different windows devices, So its purpose is an open source and cross platform compatible development tool for web apps, web APIs and microservices.

### **How to install .NET Core 3.1?**

#### **Install on Windows**

Installation options for windows can be found at:

<https://docs.microsoft.com/en-us/dotnet/core/install/windows?tabs=netcore31#install-with-power-shell-automation>

#### **Install on MacOS**

Installation options for windows can be found at:

<https://docs.microsoft.com/en-us/dotnet/core/install/macos>

#### **Install on Linux**

Installation options for Linux can be found at:

<https://docs.microsoft.com/en-us/dotnet/core/install/linux>

### **What happens with an update?**

We hope to upgrade to .NET 5 when it is launched in November, if we decide not to we will eventually have to upgrade to .NET 5 when the support for version 3.1 expires. When we upgrade we will have to update the .NET Core SDK version, update the target framework, and update package references. We will also have to review any breaking changes that the new version may bring with it.

### **How do we fix it?**

We will have to use a rolled back version of .NET Core until the necessary changes are made to the most recent version of .NET Core.

# **Test Browser**

## **Google Chrome**

### **What is Chrome?**

Google Chrome browser is an open source program for accessing the World Wide Web and running Web-based applications. The Google Chrome Web browser is based on the open source Chromium project. Google released Chrome in 2008 and issues several updates a year.

### **Why use Chrome?**

Chrome is a simple, free, and highly functional web browser with built in developer tools (DevTools) that can help us edit pages on-the-fly and diagnose problems quickly.

## **How to Install Google Chrome**

### **Install Chrome on Windows**

1. [Download the installation file.](#)
2. If prompted, click Run or Save.
3. If you chose Save, double-click the download to start installing.
4. Start Chrome:


Windows 7: A Chrome window opens once everything is done.

Windows 8 & 8.1: A welcome dialog appears. Click Next to select your default browser.

Windows 10: A Chrome window opens after everything is done. You can [make Chrome your default browser.](#)


If you've used a different browser, like Internet Explorer or Safari, you can [import your settings into Chrome.](#)

## Install Chrome on Mac

1. [Download the installation file.](#)
2. Open the file called "googlechrome.dmg."
3. In the window that opens, find Chrome .
4. Drag Chrome to the Applications folder.

You might be asked to enter the admin password.

If you don't know the admin password, drag Chrome to a place on your computer where you can make edits, like your desktop.

5. Open Chrome.
6. Open Finder.
7. In the sidebar, to the right of Google Chrome, click Eject .

## Install Chrome on Linux

Use the same software that installs programs on your computer to install Chrome. You'll be asked to enter the administrator account password.

1. [Download the installation file.](#)
2. To open the package, click OK.
3. Click Install Package.

Google Chrome will be added to your software manager so it stays up-to-date.

## What are DevTools?

### Chrome DevTools

Chrome DevTools is a set of web developer tools built directly into the [Google Chrome](#) browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.

Check out the video for live demonstrations of core DevTools workflows, including debugging CSS, prototyping CSS, debugging JavaScript, and analyzing load performance.

### How to get DevTools

Chrome DevTools is built directly into the [Google Chrome](#) browser.

### How to use DevTools

#### Open DevTools

There are many ways to open DevTools, because different users want quick access to different parts of the DevTools UI.

- When you want to work with the DOM or CSS, right-click an element on the page and select **Inspect** to jump into the **Elements** panel. Or press Command+Option+C (Mac) or Control+Shift+C (Windows, Linux, Chrome OS).
- When you want to see logged messages or run JavaScript, press Command+Option+J (Mac) or Control+Shift+J (Windows, Linux, Chrome OS) to jump straight into the **Console** panel.

See [Open Chrome DevTools](#) for more details and workflows.

## Do I want to use DevTools Extension?

### DevTools Chrome extensions

The DevTools UI is a web application embedded inside Chrome. DevTools extensions use the [Chrome extensions system](#) to add features to the DevTools. A DevTools extension can add new panels to the DevTools, add new panes to the Elements and Sources panel sidebar, examine the resources and network events, as well as evaluate JavaScript expressions in the browser tab that's being inspected.

#### If you want to develop a DevTools extension:

- If you haven't developed a Chrome extension before, see [Overview of Chrome Extensions](#).
- See [Extending DevTools](#) for the specifics of creating a Chrome DevTools extension.

For a list of sample DevTools extensions, see [Sample DevTools Extensions](#). These samples include many open source extensions that can be used for reference.

## What happens with an update?

Chrome is updated fairly regularly, at least several times per year, meaning there are plenty of chances for the browser to break. However, generally Google Chrome's updates do not break the browser for use, in the case it does we will roll back to a functional version until an update is presented.

## How do we fix it?

We can attempt to use Chrome's Beta build to get the more recently updated version, but most likely we will have to roll back to the previous version until a fix is made by Google.



## Web Server

### What is ASP.Net Cores Kestrel

#### Background:

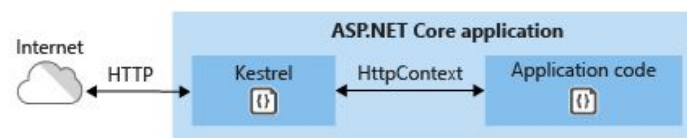
- Kestrel was launched by microsoft with ASP.NET core and is used to host ASP.NET applications on any platform.

#### Importance:

- ASP.NET core applications run on other cross-platform web servers (e.g.: NGINX and Apache) by using Kestrel as an in-process server.
- What needs to be noted is that kestrel is not a fully-featured web server.

**Therefore it is recommended to run behind web servers like IIS or NGINX.**

Kestrel used as an edge (Internet-facing) web server:



#### Pros and Cons of Kestrel:

Cons	Pros
<ul style="list-style-type: none"><li>- One of the problems with ASP.NET was the performance. Therefore, Kestrel and ASP.NET core is many times faster</li><li>- Kestrel solved the problem of getting ASP.NET to become cross-platform. So now we can write an ASP.NET core application and deploy it to Windows or linux either one.</li><li>- Kestrel is fast but lacks a lot of functionality, So it relies on a full-fledged web server to deal with things like security, management, etc. More on that later below.</li><li>- kestrel is not a fully-featured web server</li></ul>	<ul style="list-style-type: none"><li>- One of the problems with ASP.NET was the performance. Therefore, Kestrel and ASP.NET core is many times faster</li><li>- Supported Cross platform</li></ul>

**Conclusion:** We can see that a web server offers more functionality than kestrel. This is the main reason for our decision of the hosting model.

## Choosing a hosting model in ASP.NET core

### Background:

- Once our web application is successfully developed what is next? HOSTING. We have to host our application to the server so that other people can access it.
- When it comes to deployment to IIS, ASP.NET Core offers two hosting models namely InProcess and OutOfProcess.

### The Differences between the two:

<b>Hosting models:</b>	<b>In process Hosting</b>
<b>Figure:</b>	<p>The following diagram illustrates the relationship between IIS, the ASP.NET Core Module, and an app hosted in-process:</p> <pre>graph LR     Internet((Internet)) -- HTTP --&gt; ACM[ASP.NET Core Module http://contoso.com]     subgraph IIS         ACM         IISHttpServer[IISHttpServer w3wp.exe]     end     ACM &lt;--&gt; IISHttpServer     IISHttpServer -- HttpContext --&gt; AC[Application code]</pre>
<b>Definition:</b>	<ul style="list-style-type: none"><li>- IIS HTTP Server (IISHttpServer) is used instead of <a href="#">Kestrel</a> server</li></ul>
<b>Pros</b>	<ul style="list-style-type: none"><li>- For ASP.NET core 2.2 and later.</li><li>- Offers Better Performance.</li><li>- Less resource intensive as it avoids the extra network hop between IIS and Kestrel and maintaining an additional process on the machine that needs to be monitored.</li><li>- When hosting in-process, ASP.NET Core module uses an in-process server implementation for IIS, called IIS HTTP Server (IISHttpServer) and avoids the additional cost of reverse-proxying requests over to a separate dotnet process.</li></ul>
<b>Cons</b>	<ul style="list-style-type: none"><li>- Does Not implement kestrel.</li><li>- Not cross-platform.</li></ul>

<b>Hosting models:</b>	<b>Out-of process hosting</b>
<b>Figure:</b>	
<b>Definition:</b>	- <a href="#">Kestrel</a> server is used instead of IIS HTTP Server (IISHttpServer).
<b>Pros</b>	<ul style="list-style-type: none"> <li>- If you want to be 100% compatible between different deployments of the same application, whether it's on Windows or Linux, since Kestrel is the primary mechanism used to handle HTTP requests on all platforms.</li> <li>- Limits the surface area or directly facing the application to the internet.</li> </ul>
<b>Cons</b>	- Additional cost of reverse-proxying requests over to a separate dotnet process.

## Conclusion:

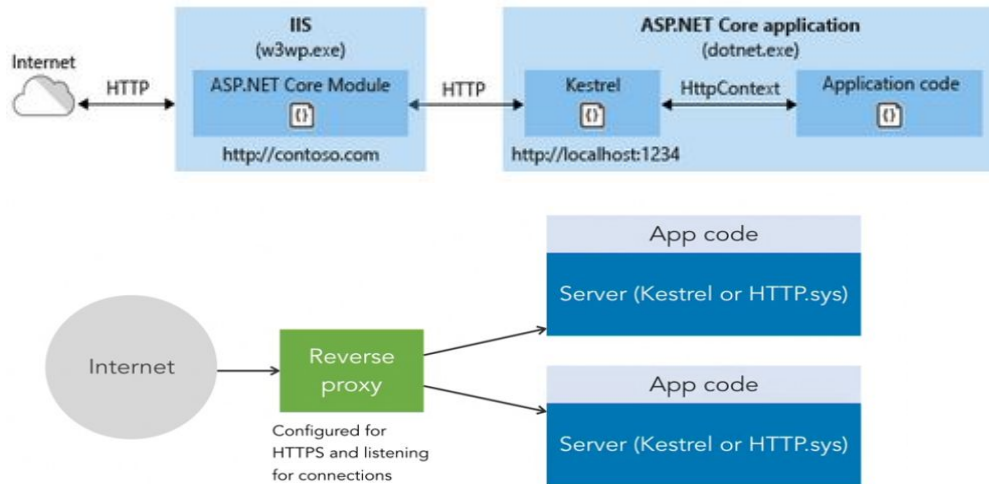
We will be using the “out-of-process” hosting model. Since we want to use different deployments between Windows and MACos of the same application and want it to be 100% compatible. We want to also add added security and scalability that comes along with putting a reverse proxy facing the internet.

## *What is a proxy server?*

A proxy server acts as a gateway between you and the internet. It's an intermediary server separating end users from our application. Below shows the benefits of using a proxy server.

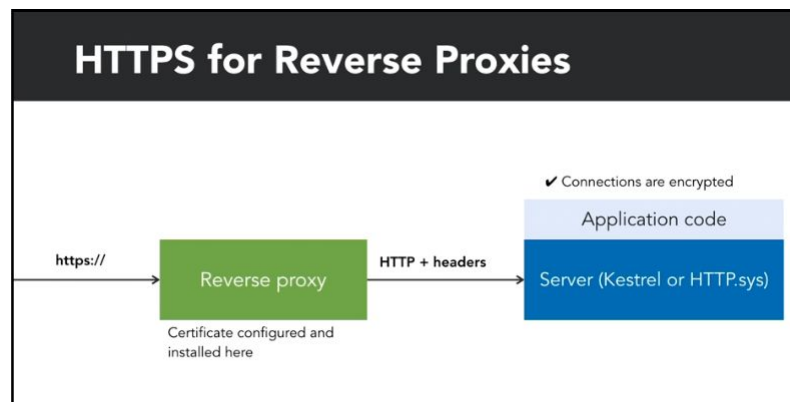


### Figures for illustration. A proxy server



### The benefits of a proxy server

- It limits your exposed surface area of the ASP.NET core application.
- A proxy server provides an additional layer of configuration and defense.
- No need to add a certificate to kestrel to enable HTTPs. A reverse proxy is a single place to configure HTTPs even if you have multiple application codes behind the proxy.



- Makes it easy to scale later, It makes load balancing simplified and send traffic to other servers and offers a secure communication as well( example: HTTPs in figure above)
- The ASP.netCoreModule running through IIS also provides process management to ensure that our application gets loaded on first access, ensures that it stays up and running and if there is a crash then restarts happens.

## Deciding on a proxy web Server to choose for our out-of-process model:

### Background:

WEB-SERVER	IIS	NGINX	APACHE
ABOUT	<p>Internet information services.</p> <p>This is a windows web server available on most versions of Microsoft's windows operating system.</p> <p>This windows service allows you to host and manage your website.</p> <p>The IIS hosts websites, web applications and services needed by the developers.</p>	<p>Nginx (pronounced Engine-X) was written specifically to address the performance limitations of Apache web servers.</p> <p>Nginx was used mostly to serve static files and as a load-balancer or reverse proxy in front of Apache installations.</p> <p>As the web evolved, and the need to squeeze every last drop of speed and hardware usage efficiency with it, more websites started to replace Apache with Nginx entirely, thanks also to a more <u>mature software</u>.</p>	<p>first released in 1995 – quickly conquered the market and became the world's most popular web server. Nowadays, it still is in that market position but mostly for <u>legacy reasons</u>.</p>

## Pros and Cons:

	PROS	CONS
<b>IIS</b>	<ul style="list-style-type: none"><li>- IIS has proven that it is easy to configure and maintain with minimal effort. (see installation steps below)</li><li>- IIS integrates really well with Visual Studio (see installation steps)</li><li>- Seamless integration with Visual Studio and dot NET applications.</li><li>- Microsoft IIS is excellent at hosting .net sites, this makes development and deployment seamless</li><li>- The SQL integration is also fairly seamless.</li><li>- IIS is faster than Apache (though still slower than nginx)</li><li>- IIS applications are using .NET framework on Windows</li></ul>	<ul style="list-style-type: none"><li>- Supported on microsoft Windows machines</li><li>- Compared to Apache or Nginx, IIS uses way more system resources.</li><li>- IIS is slower than nginx</li></ul>
<b>NGINX</b>	<ul style="list-style-type: none"><li>- Load balancing and lightweight web server which forwards the requests from the user to application server. If we host multiple sites on the same Nginx server, the load of the server doesn't increase.</li><li>- Nginx works best when we use it as a front end proxy server for any application.</li><li>- Very fast and uses few system resources.</li><li>- NGINX can also be used to improve performance because it can be used as a reverse proxy.</li><li>- NGINX has been growing rapidly</li><li>- NGINX is very fast and uses far less resources than Apache or IIS. That means it doesn't spawn new processes or threads for each Web page request. The end result is that even as the load increases, memory use remains predictable.</li><li>- Configuration of Nginx is very easy as compared to other web servers.</li></ul>	<ul style="list-style-type: none"><li>- Nginx has full support on Unix systems, but only has limited support on Microsoft Windows machines.</li><li>- Less community support compared to Apache</li></ul>

<b>APACHE</b>	<ul style="list-style-type: none"> <li>- Comes preinstalled with MACos, linux distributions</li> <li>- Great documentation and community support.</li> <li>- Documentation - Apache is the #1 web server in the world, so there is a huge amount of documentation for support</li> <li>-</li> </ul>	<ul style="list-style-type: none"> <li>- In the near future it looks like Apache will be replaced by NGINX</li> <li>- Apache is usually running PHP applications on Linux operating systems</li> </ul>
---------------	---	--

### **Conclusion:**

*Choose NGINX because of such resource efficiency and responsiveness. Also for future vision it looks like apache will be replaced by NGINX, since initially NGINX was built to address apaches performance, but then later expanded to be a full fledged web server. The only issue is that NGINX has partial support for microsoft windows, for that reason for windows systems IIS will be used. IIS has a seamless integration with .net core and it integrates well with visual studio. In conclusion: NGINX for MAC and IIS for windows*

**Which version OF IIS AND NGINX? Overreaching policy. Have a policy of updating, and look at "when does this technology stop getting support"**

- The versioning of IIS 10 and NGINX latest version 1.18.0 is chosen since we want to aim for a version that supports the following:
  - HTTP/2, a persistent connection can be used to service multiple simultaneous requests. This allows for an improvement of efficiency of HTTP over the internet.
- HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection. It also introduces unsolicited push of representations from servers to clients.

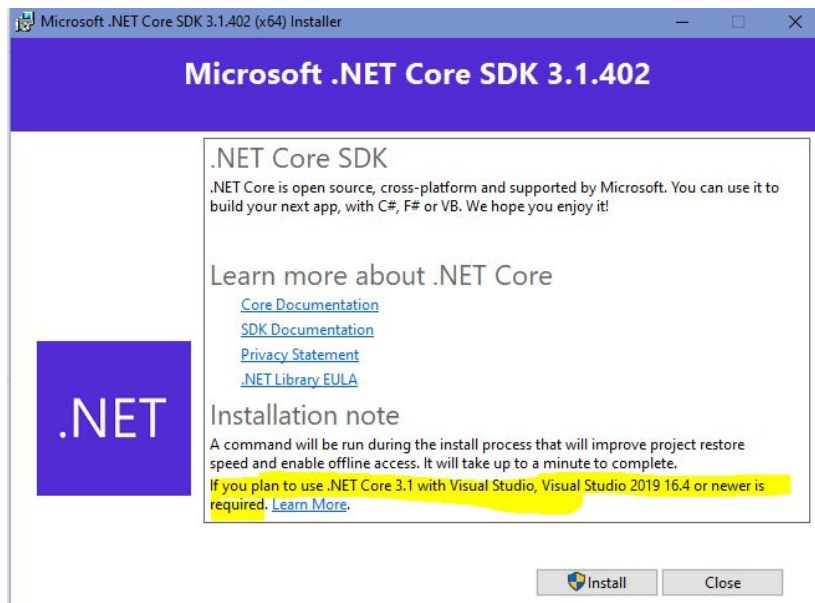
<https://www.nginx.com/blog/nginx-1-18-1-19-released/>

### **Important Post installation Steps**

- If the Hosting Bundle is installed before IIS( in the next step), the bundle installation must be repaired. Run the Hosting Bundle installer again after installing IIS

- If the Hosting Bundle is installed after installing the 64-bit (x64) version of .NET Core, SDKs might appear to be missing. So before installing bundle recommended to install the .NET core SDK from:

<https://dotnet.microsoft.com/download>



- SDK 3.1.402 has Visual Studio support : Visual Studio 2019 (v16.7) and Visual Studio 2019 for Mac (v8.7.6). SINCE This version of Visual Studio 2019 supports the latest version of Dot Net Core SDK, which is 3.0. Note that you cannot use Visual Studio 2017 to develop Dot Net Core 3 apps.
- In the normal cases, you don't have to download the SDK separately since you have installed the latest version of Visual Studio 2019, however, if you open Visual Studio 2019 and do not see the option of Dot Net Core 3 for whatever reason, then you can download the Dot Net Core SDK from the <https://dotnet.microsoft.com/download>

## Installation.

## Configuring NGINX for MAC OS

<https://rehansaeed.com/nginx-asp-net-core-depth/>

### Steps Pre-installation:

- 1) Check to see if we have apache web server on your MACOS, enter in the terminal: "httpd -v"
- 2) If you have an apache web server, stop the service first
- 3) To stop: "sudo brew services stop httpd"

- 4) Enter: "clear"
- 5) Check to see if brew is up to date: "brew update"
- 6) Installing NGINX.

### **Steps for installation:**

- 1) Use brew to install the NGINX with command: "brew install nginx"
- 2) Enter "sudo brew services start nginx" to start your new web server.

### **Steps for configuration file for NGINX :**

- 1) Locate the default place of nginx.conf file on MAC after installing with brew is:  
" /usr/local/etc/nginx/nginx.conf"
- 2) So then " sudo nano /usr/local/etc/nginx/nginx.conf" in bash
- 3) Uncomment "user nobody" and change it to your username
- 4) Save nginx.conf file
- 5) Restart the nginx server by entering: "sudo brew services restart nginx"
- 6) To verify: open your web browser, the the URL type: " localhost:8080"
- 7) Then you have successfully installed NGINX to your mac

## **Configuring IIS for Windows Systems**

### **Enable IIS**

1. Go to Start-> search control panel
2. "Programs and features"
3. "Turn windows features on or off"

OR

1. Go to Start -> search "apps and features"
2. On the right side click under related setting "Programs and features"
3. Then on the left hand side of the windows click "Turn on windows and features on or off"

Then after you followed one of the two steps above:

1. Look for "internet information services" and "IIS management tool"
2. Enable "Internet Information Services" and expand to
3. make sure "IIS management tool" is enabled as well.
4. Enable "World Wide Web Services" as well.
5. click "ok"
6. After windows completes the requested changes -> Search in the start menu for "IIS manager". (Successfully installed!)

## **.NET Core hosting bundle**

### **Why are we choosing to download the .NET Core Windows Hosting Pack?**

- SDK and Runtimes by themselves are usually not the right tool for deployed applications because they don't include everything you need and for this reason - at least on Windows there's a special Hosting Pack download that contains everything you need to run an ASP.NET Core application on Windows.
- What it contains:
  - Everything you need to run an ASP.NET core application on WINDOWS
  - Does not include dotnet SDK tools (IMPORTANT: see 9 Post installation steps.)
  - This package installs both 32 and 64 bit runtimes, the ASP.NET Core Runtimes as well the IIS hosting components on Windows.

### **Install .NET Core hosting bundle**

Steps:

- 1) Go to -> <https://dotnet.microsoft.com/download/dotnet-core>
- 2) choose the .NET Core 3.1 (recommended) version
- 3) In the Run apps - Runtime column, ASP.NET Core runtime 3.1.8"
- 4) Download the installer using the Hosting Bundle link "Hosting bundle"
- 5) Agree to the license terms and click install

How to verify: C:\Program Files\dotnet\shared\Microsoft

### **Restart IIS**

Two options:

- 1) Restart machine
- 2) Restart IIS alone-> open command prompt as administrator -> " run "iisreset"

```
C:\WINDOWS\system32>iisreset

Attempting stop...
Internet services successfully stopped
Attempting start...
Internet services successfully restarted

C:\WINDOWS\system32>
```

### **Demonstrate IIS to host asp.net core application**

- 1) On the hosting system, create a folder to contain the app's published folders and files.

- First I created a folder called websites and a subdirectory called HelloCoreWorld as such: C:\Websites\HelloCoreWorld. This is the physical path where the files and binaries are going to live for the website created.
- 2) In windows 10 taskbar search “IIS manager”
  - 3) In IIS Manager, open the server's node in the Connections panel. Right-click the Sites folder. Select Add Website from the contextual menu.
  - 4) Provide a Site name and set the Physical path to the app's deployment folder. Provide the Binding configuration and create the website by selecting, like so:

The screenshot shows the 'Add Website' dialog box with the following configuration:

- Site name:** HelloCoreWorld
- Application pool:** HelloCoreWorld
- Content Directory:**
  - Physical path:** C:\Websites\HelloCoreWorld
  - Pass-through authentication:** (unchecked)
- Binding:**
  - Type:** http
  - IP address:** All Unassigned
  - Port:** 80
  - Host name:** localhost
- Start Website immediately:** (checked)

An arrow points to the 'OK' button at the bottom right of the dialog.

- 5) I used Host name: localhost” since this is going to be local testing
- 6) Now we need to modify the setting for the “Application Pool” for .net core:
- 7) Left click on “Application pools” left click on the Application pool just created in our case it is “HelloCorreWorld” and click “Basic setting”
- 8) For the “.NET CLR visions” click “No managed code” .
  - a) ASP.NET Core runs in a separate process and manages the runtime. ASP.NET Core doesn't rely on loading the desktop CLR (.NET CLR). This says the .NET code will be run and managed by Kestrel.
  - b) For the Dot Net Core Apps to work under IIS, we will have to create a new application pool with the no managed code option. The IIS Application pool will not have any effect on the runtime of the Dot Net Core Apps, it only works as a reverse proxy



## Verify:

- 1) Go to "Sites" under "Connections" tab
- 2) click "HelloCoreWorld"
- 3) On the right hand side panel under "Browse Website" click " Browse localhost on \*:80(http).
- 4) Result: will show an error since currently there are no files in that directory which is okay. This shows that IIS is running and the setup is complete. (will fix this error in a bit)

### HTTP Error 403.14 - Forbidden

The Web server is configured to not list the contents of this directory.

#### Most likely causes:

- A default document is not configured for the requested URL, and directory browsing is not enabled on the server.

#### Things you can try:

- If you do not want to enable directory browsing, ensure that a default document is configured and that the file exists.
- Enable directory browsing using IIS Manager.
  1. Open IIS Manager.
  2. In the Features view, double-click Directory Browsing.
  3. On the Directory Browsing page, in the Actions pane, click Enable.
- Verify that the configuration/system.webServer/directoryBrowse@enabled attribute is set to true in the site or application configuration file.

#### Detailed Error Information:

<b>Module</b>	DirectoryListingModule	<b>Requested URL</b>	http://localhost:80/
<b>Notification</b>	ExecuteRequestHandler	<b>Physical Path</b>	C:\Websites\HelloCoreWorld
<b>Handler</b>	StaticFile	<b>Logon Method</b>	Anonymous
<b>Error Code</b>	0x00000000	<b>Logon User</b>	Anonymous

#### More Information:

This error occurs when a document is not specified in the URL, no default document is specified for the Web site or application, and directory listing is not enabled for the Web site or application. This setting may be disabled on purpose to secure the contents of the server.

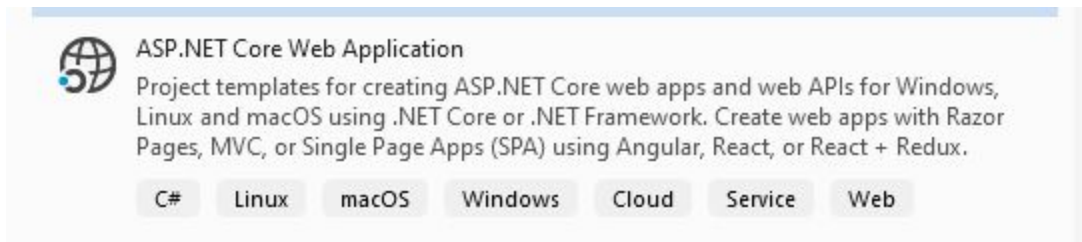
[View more information >](#)

## Deploying website using visual studio

### Steps:

- 1) Deploy the app to the IIS Physical path folder that was established in the Creating the IIS site section.
- 2) Web deploy with Visual studio community 2019:
- 3) Go to File
- 4) Select new project

## 5) Select Template: “ASP.NET core Web Application”



6) Click next

7) Project name : “HelloCoreWorld”

8) Location is the same directory as the one used in the previous section  
“C:\Websites\HelloCoreWorld. “

9) Click “create application”

10) Click “Web application (Model-View-Controller)”

11) Make sure “.NET core” and “ASP.NET core 3.1” are selected

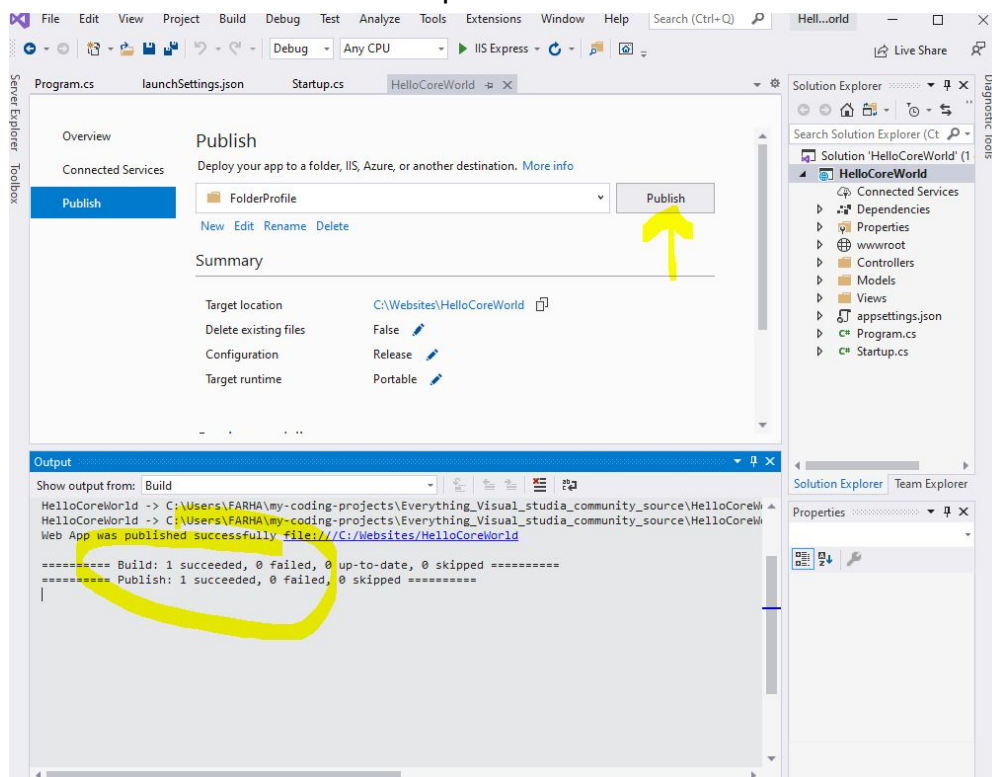
12) On the left side, The Solution Explorer column

13) Left click on “HelloCoreworld” project and select “publish”

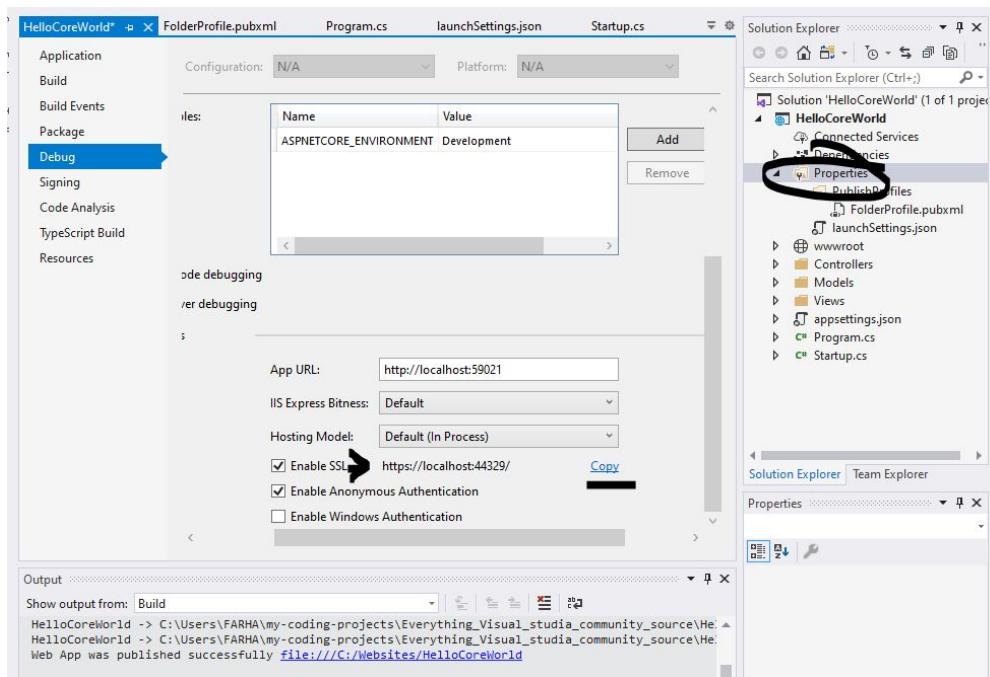
14) Select “Delete existing files”

15) Under “File publish options” check off “Delete all existing files prior to publish”

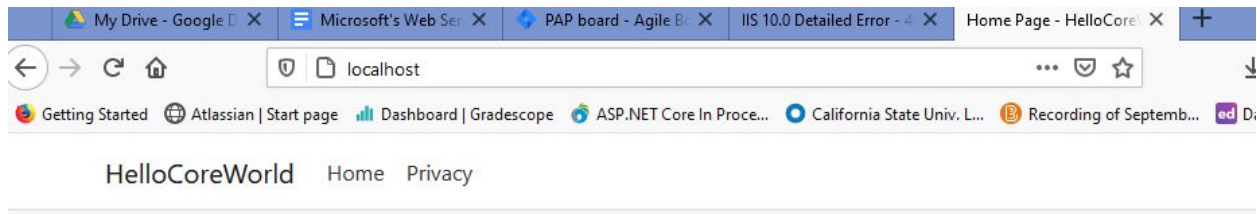
16) Press ok the click “Publish” output should show:



17) Then In the solutions column go to Properties and copy "App URL"



18) Then search:



That is how you successfully publish your application to IIS using visual studio.

## **5 Appendices**

### **References:**

- <https://stackify.com/what-is-kestrel-web-server/>
- <https://kinsta.com/blog/nginx-vs-apache/>
- <https://www.zdnet.com/article/nginx-takes-2nd-place-in-web-servers-from-microsoft-iis/>
- <https://www.zdnet.com/article/apache-and-iis-web-server-rival-nginx-is-growing-fast/>
- <https://www.trustradius.com/products/iis/reviews>
- <https://www.trustradius.com/products/nginx/reviews>
- <https://www.plesk.com/blog/various/nginx-vs-apache-which-is-the-best-web-server/>
- <https://serverguy.com/comparison/apache-vs-nginx/>
- <http://www.devx.com/webdev/performance-comparison-apache-vs.-iis.html>
- <https://stackshare.io/microsoft/microsoft>
- <https://www.quora.com/Why-isn%E2%80%99t-the-Microsoft-Stack-as-popular-as-other-web-development-stacks><http://www.xavierdilipkumar.com/post/what-are-lamp-wamp-mamp-and-wisa-stacks>
- <https://medium.com/@aram161287/deploy-asp-net-core-web-api-on-iis-f75e755a6402>
- <https://www.youtube.com/watch?v=Sw6USmvt60s>
- <https://www.youtube.com/watch?v=fQ7BzRlr56M>
- <https://www.youtube.com/watch?v=mtXE1LMQZEY>
- [https://www.youtube.com/watch?v=H7wZ2khnT\\_E](https://www.youtube.com/watch?v=H7wZ2khnT_E)
- <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-2.1&tabs=aspnetcore2x#data-protection>