Introduction to Visual Studio 2017

MD. MAHEDEE HASAN

MAHEDEE.NET

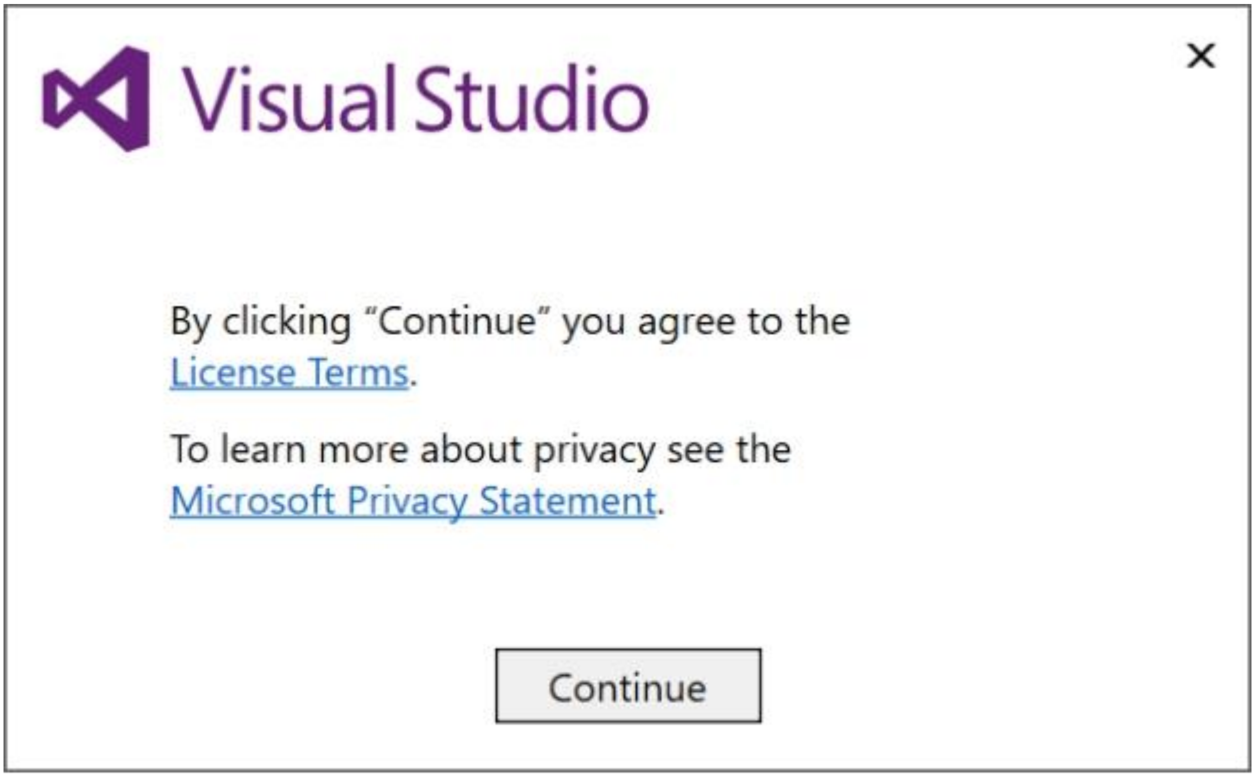
Introduction to Visual Studio 2017

Md. Mahedee Hasan
Microsoft Most Valuable Professional (MVP),
Artificial Intelligence (AI)
Microsoft Technology Specialist (MCTS)
Senior Software Architect
Leadsoft Bangladesh Limited
Microsoft MVP | Technical Speaker | Technical Blogger | MCTS
Trainer, LEADS Technology Ltd.
Blog: http://mahedee.net/
Email: mahedee.hasan@gmail.com
Linkedin: http://www.linkedin.com/in/mahedee
***TechNet*: https://social.technet.microsoft.com/profile/mahedee/**
***Facebook*: http://www.facebook.com/mahedee19**

Contents

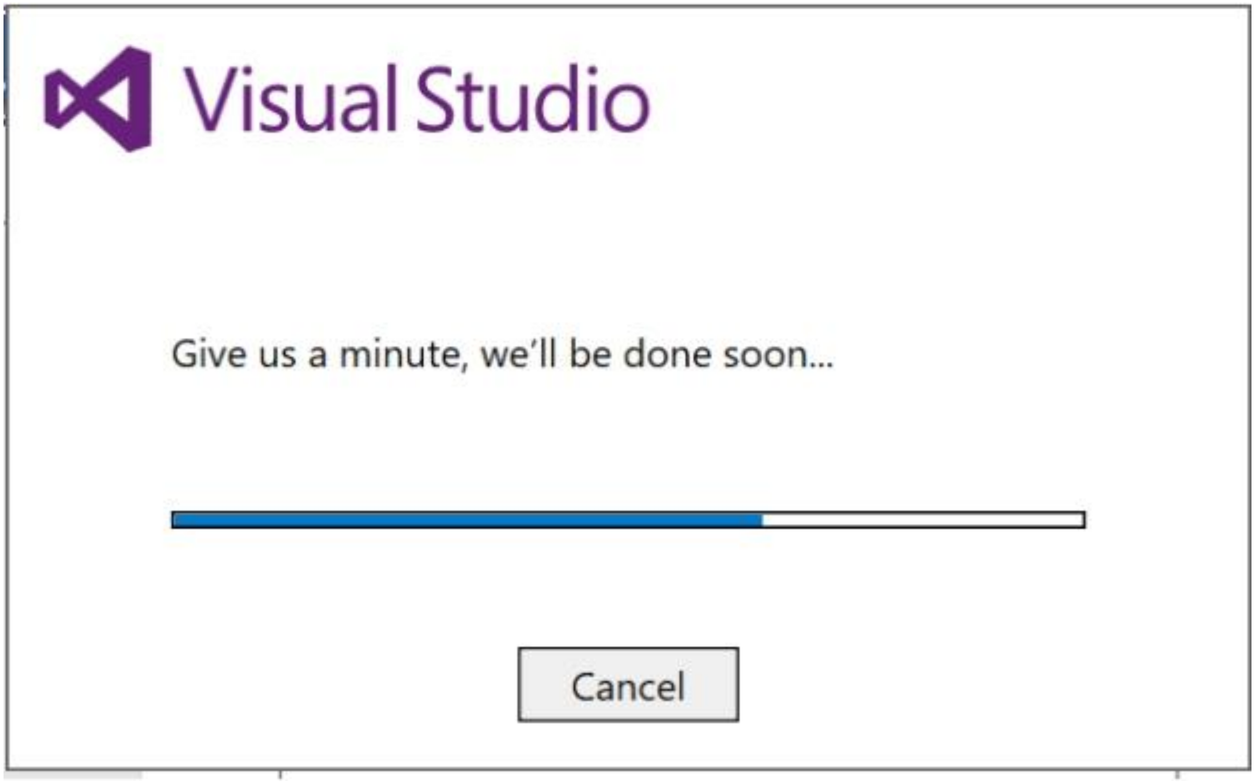
Installing Visual Studio 2017

- Enable internet connection
- Run the exe file and you will see a very unique launch screen, as shown below



By clicking "Continue" you agree to the License Terms.

To learn more about privacy see the Microsoft Privacy Statement.

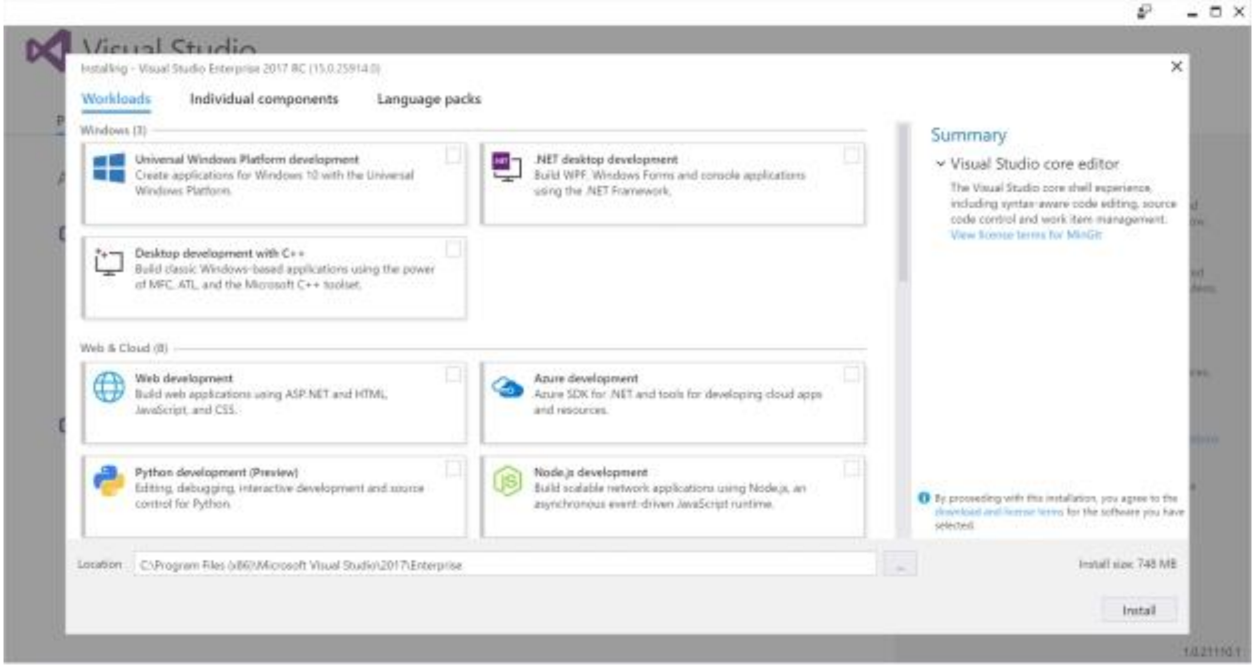


Give us a minute, we'll be done soon...

Option Selection

Visual Studio 2017 offers a brand new way of choosing the tool sets. You can choose one of the following.

1. Workload; i.e., what exactly you do or want to do.
2. Individual components; i.e., hand pick all the components individually. I feel it can be messy, unless you want to cherry pick.
3. Language Packs; i.e., select language of your choice. Ex. Choose .NET desktop development and Web development



Installing - Visual Studio Enterprise 2017 RC (15.0.25914.0)

Windows (3)

Universal Windows Platform development
Create applications for Windows 10 with the Universal Windows Platform.

.NET desktop development
Build WPF, Windows Forms and console applications using the .NET Framework.

Desktop development with C++
Build classic Windows-based applications using the power of MFC, ATL, and the Microsoft C++ toolset.

Web & Cloud (8)

Web development
Build web applications using ASP.NET and HTML, JavaScript, and CSS.

Azure development
Azure SDK for .NET and tools for developing cloud apps and resources.

Python development (Preview)
Editing, debugging, interactive development and source control for Python.

Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Visual Studio core editor
The Visual Studio core shell experience, including syntax-aware code editing, source code control and work item management.
View license terms for MinGit

By proceeding with this installation, you agree to the download and license terms for the software you have selected.

Location C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise

Install size: 748 MB

1.0.21110.1

- Then, basically next, next, next
- Takes about 1 hour depending on your internet speed.

Demo: Installing Visual Studio 2017

Projects and Solutions

- Each **independently buildable** thing in Visual Studio is a **"project"**
  - May build to an **EXE or a DLL**
  - Contains code in a **single language** (C#, VB, C++, etc)

- **Several projects** may be **combined** into a "**solution**"
  - The solution is what you open in Visual Studio
  - You can **build the whole solution** at once

- **Options and settings** apply at **different levels**
  - To your copy of **Visual Studio**
    - Eg font choices, tool pane layouts
  - To the **solution**
    - Eg which project to launch when you start a debug session
  - To the **project**
    - Eg compiler options

Demo: Project and Solution

Application Types

- Visual Studio can build a **wide variety** of applications
- **Native or managed**
  - **Uses framework or not** for run time
- Client Windows
  - Desktop WPF, Windows Forms, MFC, …
  - Windows Phone
  - Windows Store

- Web ASP.NET WebForms
  - ASP.NET MVC
  - SharePoint web parts

- Many more
  - Console application
  - Windows service
  - Web service
  - Classic server

Demo: Application Types

Settings

- Visual Studio settings apply to all the solutions you open
- There are a number of **"preset" settings**
  - C++
  - **C#**
  - VB etc.

- You can **export and import** settings
  - Share with your team
  - Switch to "presentation mode" and back

Demo: Settings

Visual Studio Versions and SKUs

- https://visualstudio.microsoft.com/
- Visual Studio Community 2017
  - Completely free and no royalties, can create commercial applications
  - Simplified UI
  - Some functionality not included
- Visual Studio Professional 2017
  - Features for a solo developer
- Visual Studio Enterprise 2017
  - Include all features

Visual Studio at a glance

- Visual Studio is an **Integrated Development Environment** (IDE)
  - **Editor**
  - **Compiler**
  - **Debugger**
  - Much more

- What you open in Visual Studio is a **solution**
  - Holds one or more **projects**

- A project is what you build
  - Some projects can be **executed directly**
  - Others are **called by a hosting process** or provide a library

Demo: Create a Console Application

Demo: Create a ASP.NET Windows Application

Demo: Create a ASP.NET Web form Application

Demo: Create a ASP.NET MVC Web Application

Using Namespaces

- **Referring to code in another assembly**, use the full name of a class

  - System.Windows.Window
  - GreetingLogic.Greeting

- As a convenience, can “**use**” or “import” the namespace:
- using System.Windows;
  - … Window …
- using GreetingLogic;
  - … Greeting …
- **Adding a reference** doesn’t automatically use the namespace
- Using the namespace doesn’t **automatically** add the reference

Folders and Subfolders

- Folder for solution
  - Holds .**sln**, .**suo**
- One folder under it for each project
  - Holds .**csproj**, all code files
- Folder under project called **bin**
  - Short for **Binaries**
  - Typically has **Debug and Release folders**
  - These hold exe or dlls
- Other folders under project
  - Obj: used by **build system**, **can ignore**
  - Properties: Visual Studio keeps project properties and code you don’t normally edit
- **Startup project has all the build outputs**
  - **Copied from other projects**

Demo: Folders and Subfolders

Architecture

- **Design** your **solution** before opening Visual Studio
  - What **projects** will you add?
  - What will you **name** them?
  - Is it **OK for the solution** to have the same name as the first project?
    - If not, create an empty one and add projects, or rename after you create

- **Which projects will refer** to which?
  - Avoid **circular** references
  - Think about where to **put shared logic**
  - May need to create an extra project to **hold utilities**

- Get the structure in place and then add code
- Experienced developers know where everything is kept

    - Visual Studio will help you find them
    - Visual Studio lets you change almost everything
    - Sticking to the **usual defaults** makes life simpler for the rest of your team

Demo: Architecture

History Card

Version No	Modification History	Update Date	Published Date
1	Created	2014-11-01	2014-11-24
2	Updated for VS 2013	2016-02-02	2016-02-02
3	Updated for VS 2017	2018-07-31	2018-07-31