**Hands-on .NetCore 3.0**

Table of Contents

[Introduction 2](#_Toc21981816)

[Things covered during this course 2](#_Toc21981817)

[What will you need? 2](#_Toc21981818)

[Getting Started with .Net Core 2](#_Toc21981819)

[Installing VS Code 2](#_Toc21981820)

[Installing .Net Core 3](#_Toc21981821)

[Install Extension 4](#_Toc21981822)

[Helpful links 5](#_Toc21981823)

[Base folder to house all the exercises 5](#_Toc21981824)

[Test App - To check everything is working 5](#_Toc21981825)

[Create Test App 5](#_Toc21981826)

[Run the App 6](#_Toc21981827)

[Scaffolding Applications with .Net CLI 7](#_Toc21981828)

# Introduction

This is an extensive course which will cover .NetCore 2.2 and will build on concepts learned while doing [DI-ConfigSettings-Logging-NetCore2.2](https://itplate.blogspot.com/2019/10/di-configsettings-logging-netcore22.html); but is not pre-requisite.

This course is meant to be hands on, so follow the instructions. Learn to build Console, MVC, and Web API Applications in Windows, Mac, or Linux with the latest version of .NET Core.

## Things covered during this course

* Install .NetCore
* Configure our tools – VS Code
* Setup .Net Core on Servers (optional)
* Master the .Net Core CLI
* Write Console Applications
* Build ASP.Net Core Web Applications
* Build ASP.Net Core Web API Applications

## What will you need?

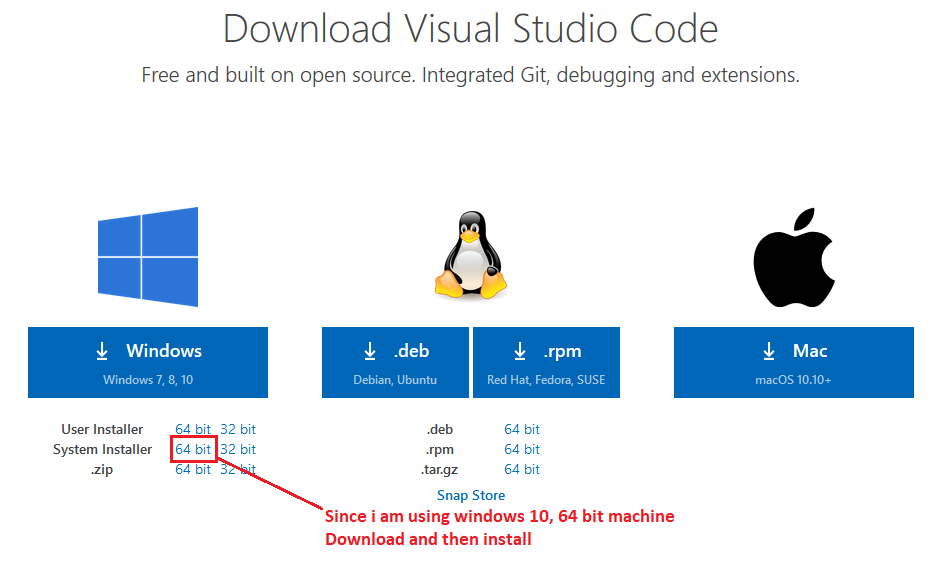
* Windows 10, OSX or Linux
* Visual Studio Code, [download link](https://code.visualstudio.com/download)
* .Net Core SDK
* Visual Studio Community Edition 2017/2019 (Free) to cover some of the C#7 examples, [download link](https://visualstudio.microsoft.com/downloads/)

# Getting Started with .Net Core

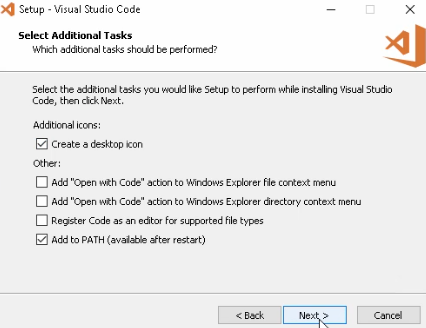
We’ll setup the environment in this section. I’ll be using a windows machine but installing the CLI and VS Code is the same.

## Installing VS Code

Download link: <https://code.visualstudio.com/download>



After download double click to install

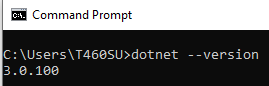


Finish the installation

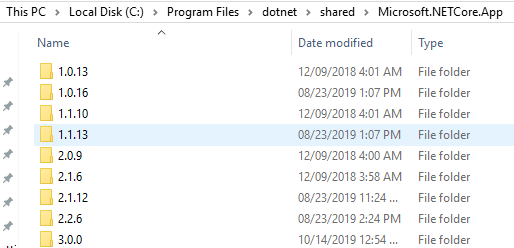
## Installing .Net Core

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

Bring up the command prompt and then check with command **dotnet –version**

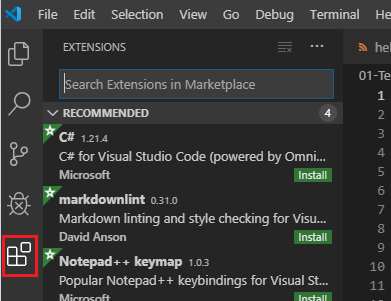


Another way to check all the versions is to navigate to: **C:\Program Files\dotnet\shared\Microsoft.NETCore.App**



Even though I have version 3.0 installed, for this hands on course, I will be using 2.2.

## Install Extension

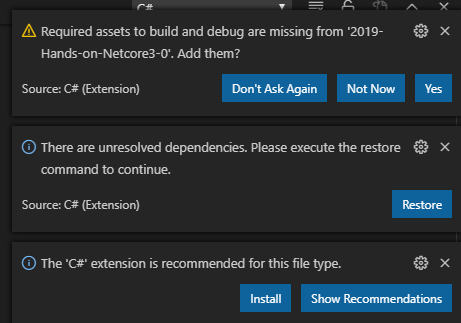


Click Extensions tab.

Then install following extensions

* Search for C# and install “C# for Visual Studio Code (powered by OmniSharp) by Microsoft”
* Search for Core and install “.NET Core Starter's Pack by Blair Leduc”. Click on “Activate” to activate “Material theme”
* Search for and install “C# FixFormat by Leopotam”

Also get the recommendations

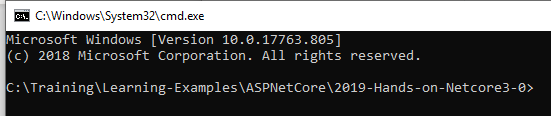


# Helpful links

Command line interface: <https://www.tutorialsteacher.com/core/net-core-command-line-interface>

# Base folder to house all the exercises

Create a base folder on your C drive which will all the hands-on exercises.

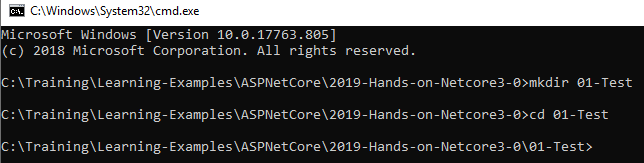


At this point you can type **code .** (code space dot) to open the base folder in VS Code.

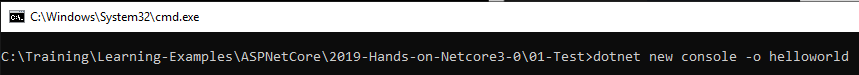
# Test App - To check everything is working

## Create Test App

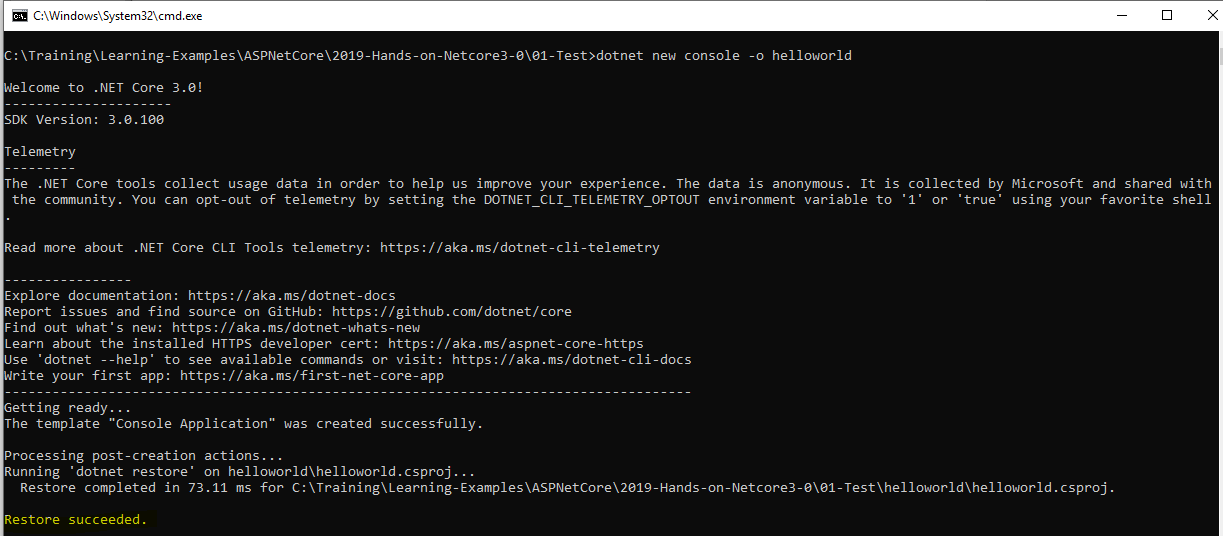
Create a new sub folder 01-Test and switch to it



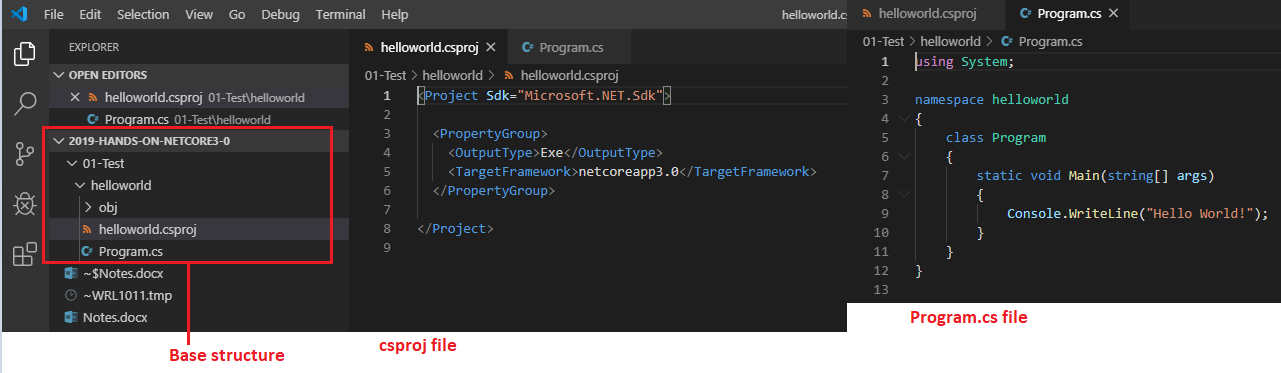
Here we’ll create a new console app



A new console app will be created inside the **01-Test** folder with name **helloworld**.

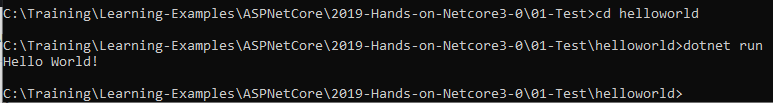


Since we have already opened the [base folder in VS Code](#_Base_folder_to), the new structure should show there.



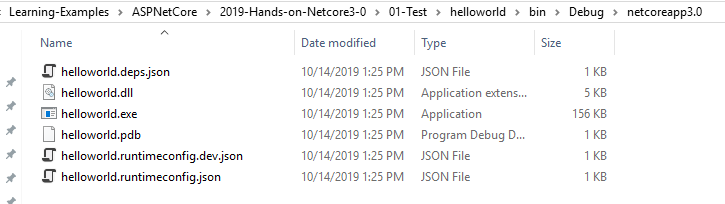
## Run the App

* Switch to the **helloworld**
* And then use command **dotnet run**



We should see a message **Hello World!**

Here is our application



# Scaffolding Applications with .Net CLI

**Basics**

* dotnet restore : pulls in the dependencies needed by the application
* dotnet run : compiles and run the application
* dotnet build : compiles the application
* dotnet publish : packages up the files for reuse

**Scaffolding**

* dotnet new console -o ConsoleTest
  + dotnet run (to run the application)
  + dotnet build --force
  + dotnet build --no-restore
  + to add reference to the below classlib
    - dotnet add reference “..\StringModifier.csproj”
* dotnet new classlib -o StringModifier
  + dotnet build
* dotnet new mstest -o libraryTester
  + add a reference to the abpove StringBuilder library
    - dotnet add reference “..\StringModifier.csproj”
    - create UnitTest.cs , reference the above and then test
  + dotnet build
  + dotnet test
  + if [TestCategory(“SomeThing”)] applied to the tests then these could be run as
    - dotnet test --filter TestCategory=SomeThing
* dotnet new mvc -o mvctest
  + dotnet run
    - will give you a port to run the app like
      * http://localhost:5000
  + dotnet build
  + authentication
    - dotnet new mvc -o mvcauth individual
* dotnet new webapi -o apidemo
  + can add authentication just like above
  + dotnet run
    - this will give you the port
      * http://localhost:5000
* dotnet new angular -o ngdemo
  + install node js and then run npm
    - npm install
  + dotnet run
    - will give the port number
      * http://localhost:5000
* Other functions can that could be done via Scaffolding, check online for documentation
  + Razor views
  + ReactJs
  + xUnit test project
  + Empty ASP.Net Core Apps
  + Nuget packages
  + MVC with no views