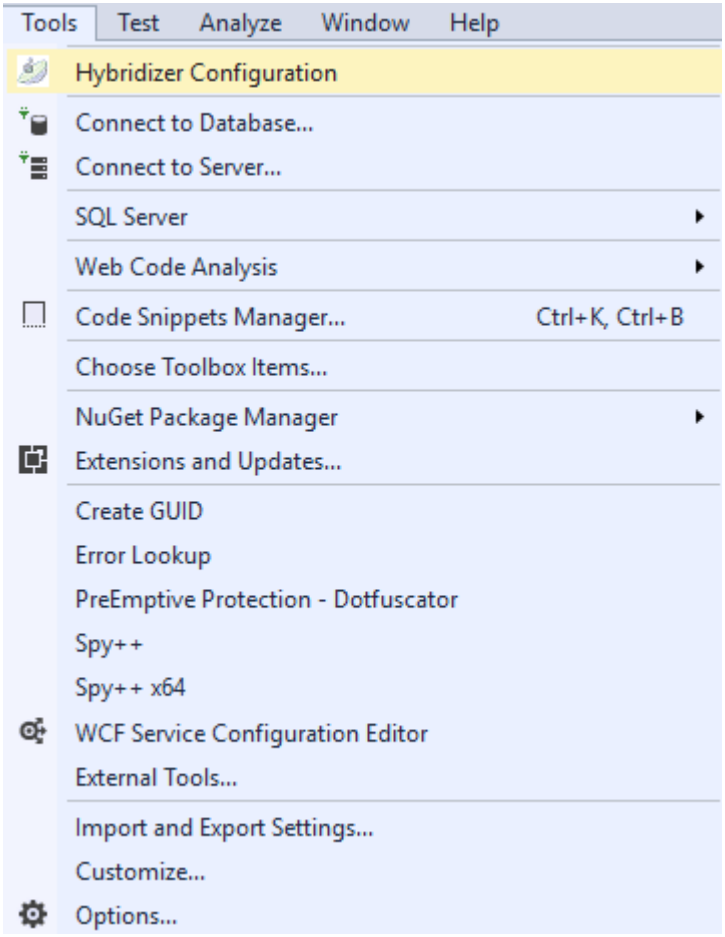
Hybridizer Basic Getting Started Guide

This short tutorial details how to get started with Hybridizer Basic.

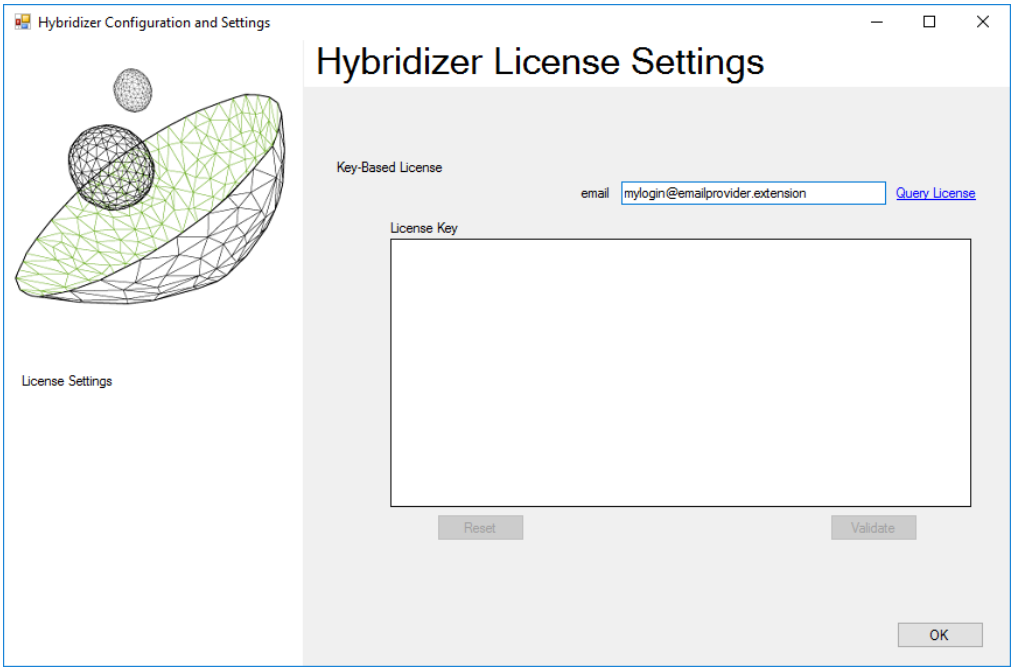
1. Install the VSIX package (if not already installed)

2. Query a license for Hybridizer Basic:

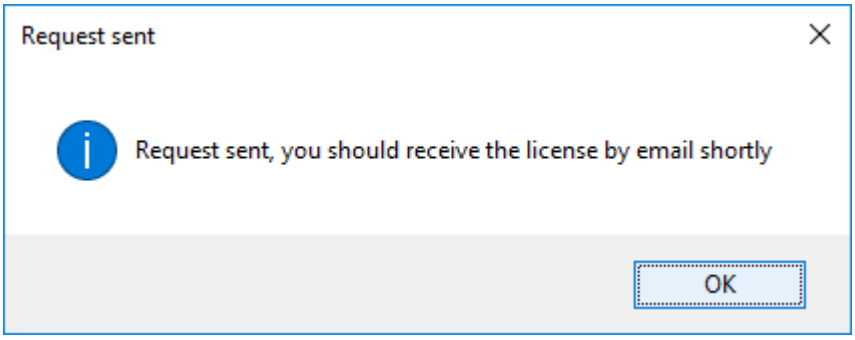
Go to Tools\Hybridizer Configuration



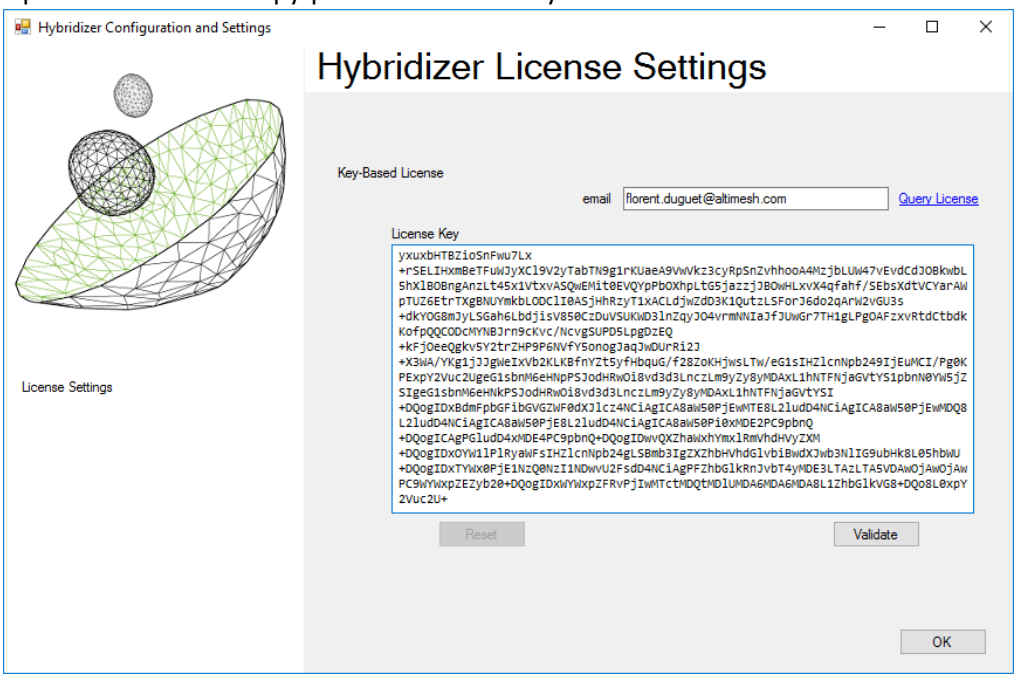
Type-in your email in the appropriate box, and click Query License



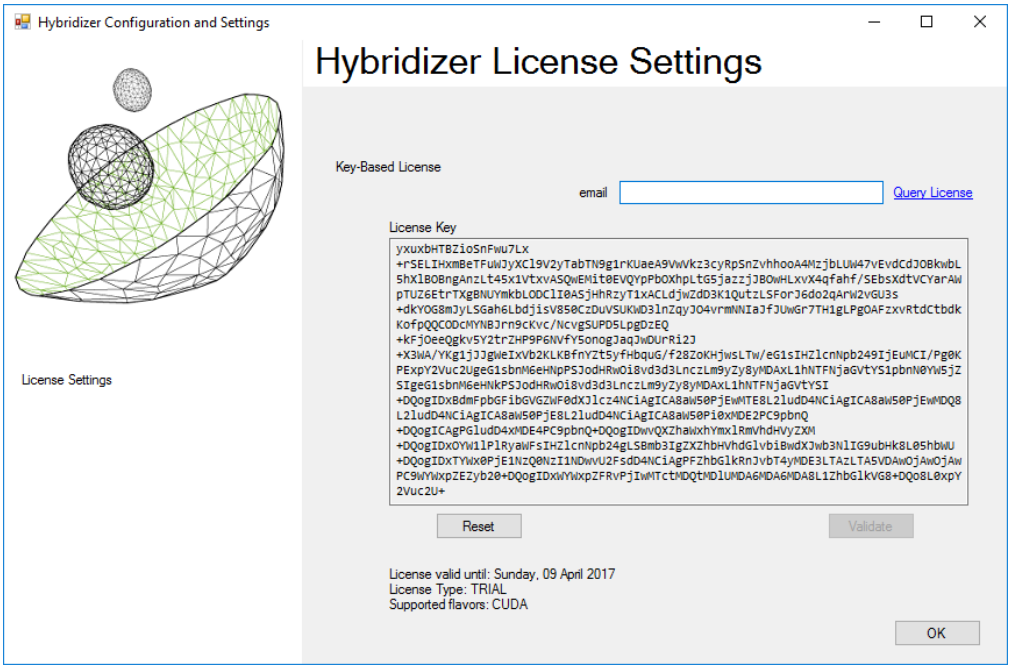
The following pop-up message should appear:



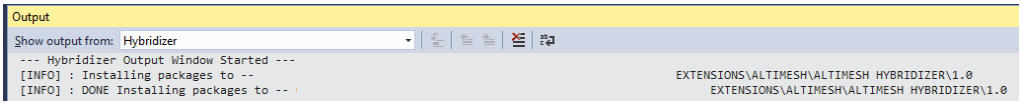
Open the email and copy-paste the license key in the text-box.



Click Validate to finalize set-up.

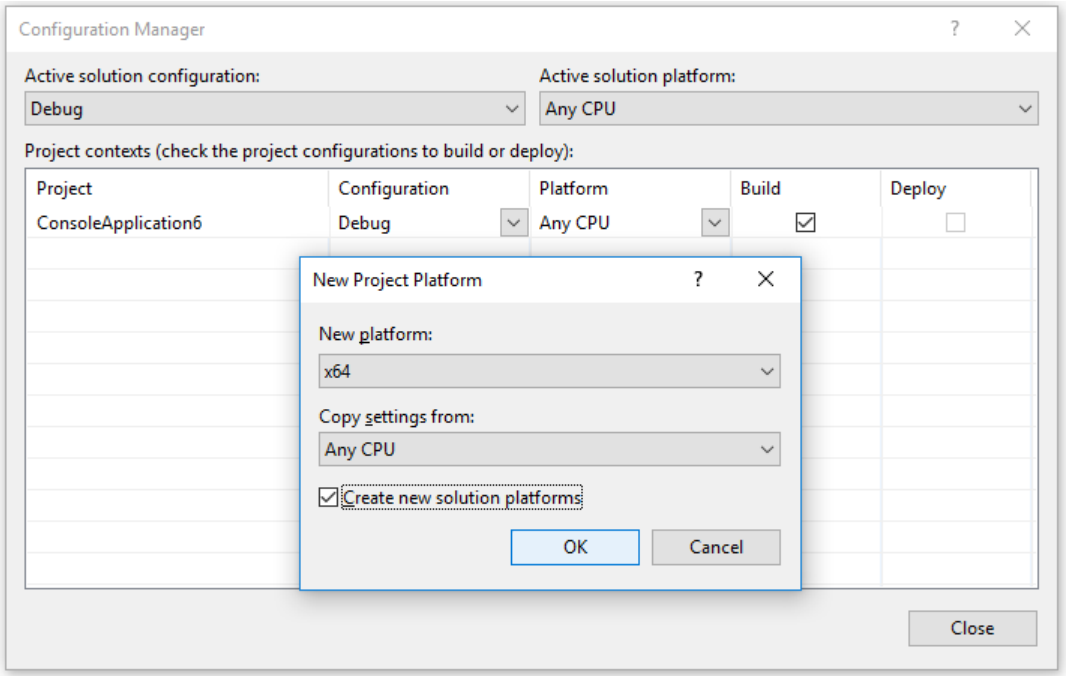


Some description of the package should appear in the lower part of the frame.
Verify that the output window, section Hybridizer, the packages have been installed in the extension location:

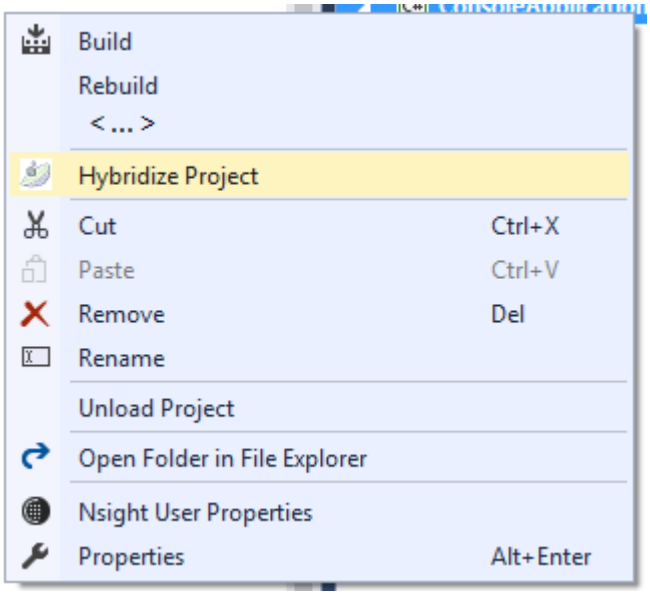


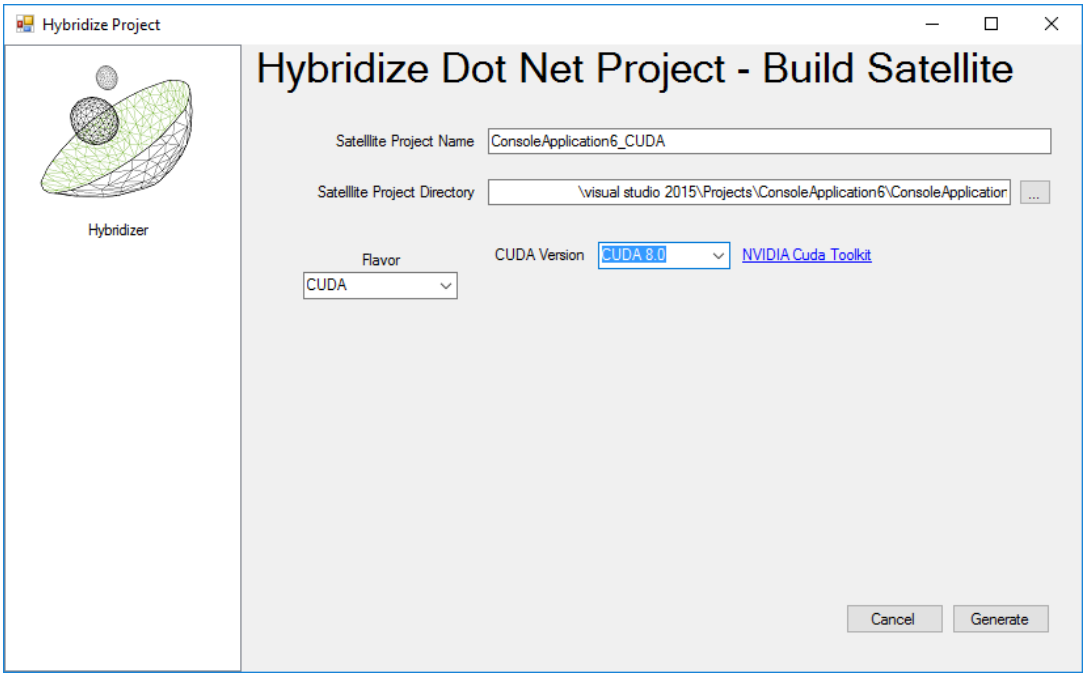
3. Create a new CSharp project, Console application.

4. Create and select x64 platform for the solution and the project:



5. Right-click project in Solution Explorer and select Hybridize Project





Then, click Generate to generate the project within the solution for the CUDA flavor (Here with CUDA 8.0).

Doing so, the CSharp project will have a new reference (Hybridizer.Runtime.CUDAImports), and the CUDA project will be generated.

6. Write a kernel:
Within the program class, declare a public static method as follows:

```
[EntryPoint("run")]
public static void Run(int N, int[] a, int[] b)
{
    Parallel.For(0, N, i => { a[i] += b[i]; });
}
```

The EntryPoint attribute indicates that this method should be runnable on GPU.

7. Call the kernel:
   In the body of the main method:

```
static void Main(string[] args)
{
    int[] a = { 1, 2, 3, 4, 5 };
    int[] b = { 10, 20, 30, 40, 50 };

    // create an instance of HybRunner object to wrap calls on GPU
    HybRunner runner = HybRunner.Cuda("ConsoleApplication6_CUDA.dll");

    // create a wrapper object to call GPU methods instead of C#
    dynamic wrapped = runner.Wrap(new Program());

    // run the method on GPU
    wrapped.Run(a.Length, a, b);

    // verify the results
    for (int k = 0; k < a.Length; ++k)
    {
        if (a[k] != (11 * (k + 1)))
            Console.Out.WriteLine("ERROR !");
    }
    Console.Out.WriteLine("SUCCESS");
}
```

8. Build the solution, and run the CSharp application.

   The output should print the following:

```
Using CUDA 80
[INFO] : Registered DLL <path-to>\ConsoleApplication6_CUDA.dll
SUCCESS
Press any key to continue . . .
```