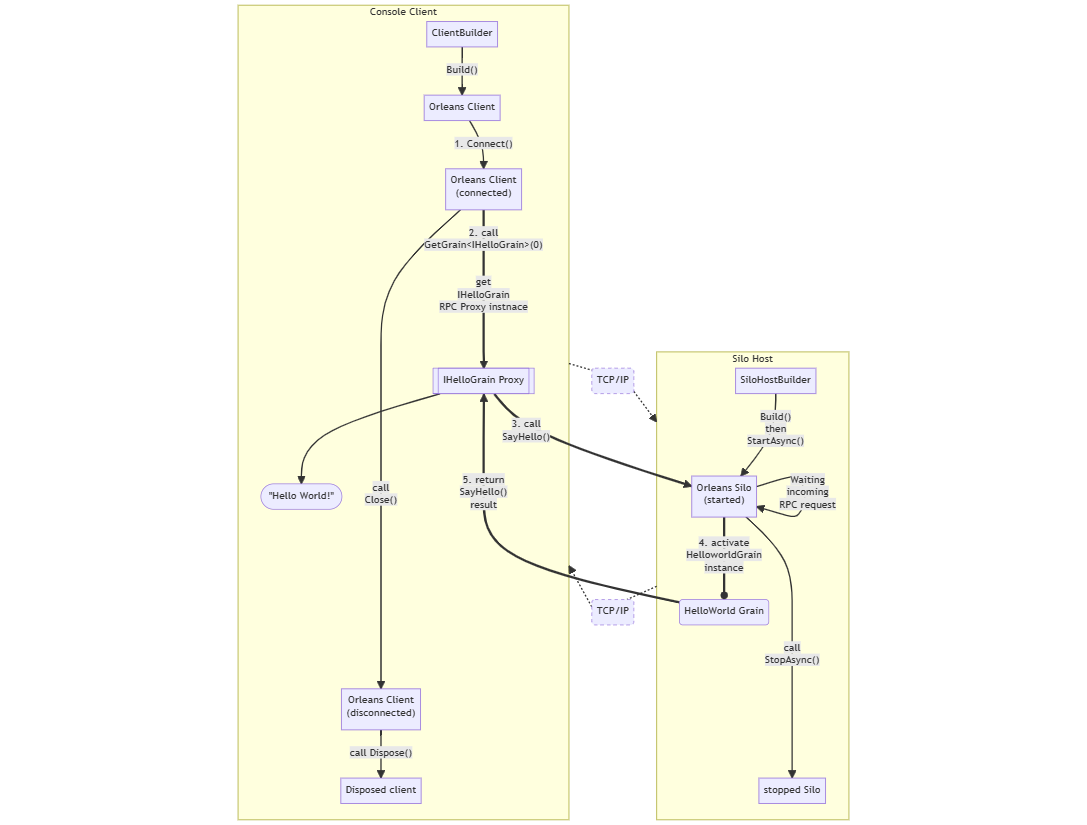
# 第一個Helloworld專案的建置與執行

## HelloWorld Grain的執行架構

HelloWorld專案程式碼完成後的執行架構將如下圖所示：

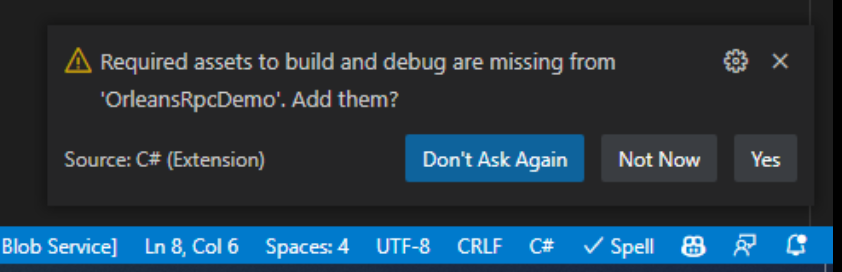


此為Client端和Silo後端各自從程式起動到結束的執行流程，其中最重要的部分是Client端呼叫Silo後端Grain RPC方法的過程：

1. Client端呼叫Orleans Client端物件的Connect()方法取得和Silo後端的TCP/IP連線。
2. Client端呼叫Orleans Client端物件的GetGrain<T>()方法取得Grain在Client端的RPC Proxy實例。
3. Client端呼叫Grain RPC Proxy實例的RPC方法SayHello()，此時Grain RPC Proxy實例會將原本C#的非同步呼叫轉換成RPC方法的底層呼叫機制，透過TCP/IP網路傳送呼叫參數給距離最近的Silo後端。
4. Silo後端接收到RPC呼叫後，會根據呼叫參數建立HelloGrain實例，並呼叫Grain實例的RPC方法實作。
5. Grain的RPC方法實作完成後，會將結果傳回Silo後端，然後Silo後端再藉由TCP/IP網路將結果傳回給還在非同步方法等待完成的Client端，此過程由於繁雜且實際上Orleans框架底層都幫你做完了，因此在圖上的流程就直接抽象化理解為Grain非同步方法完成後透過TCP/IP網路回傳結果給Client端RPC Proxy。

雖然圖中是畫成Client和Server端是靠TCP/IP網路連接，但但後面示範的程式實際上Client和Server都是跑在同台電腦用Localhost的方式連接：Client端的ClientBuilder程式碼會呼叫 [UseLocalhostClustering()](https://learn.microsoft.com/dotnet/api/orleans.clientbuilderextensions.uselocalhostclustering) 擴充方法以便稍後呼叫 Connect() 時連接本機Silo，而Server端的SiloHostBuilder程式碼也有一個 [UseLocalhostClustering()](https://learn.microsoft.com/en-us/dotnet/api/orleans.hosting.corehostingextensions.uselocalhostclustering) 擴充方法便配置使用Local的測試用Silo配置。

## 專案建置步驟

1. 將[前天完成檔案結構的HelloWorld Grain專案進度](https://github.com/windperson/OrleansRpcDemo/tree/day04)，安裝上對應的Nuget套件：
   * **RpcDemo.Interfaces.Hello**專案：RPC介面專案，安裝套件有：
     + [Microsoft.Orleans.Core.Abstractions](https://www.nuget.org/packages/Microsoft.Orleans.Core.Abstractions)
     + [Microsoft.Orleans.CodeGenerator.MSBuild](https://www.nuget.org/packages/Microsoft.Orleans.CodeGenerator.MSBuild)
   * **RpcDemo.Grains.Greeting**專案：Grain實作專案，安裝套件有：
     + [Microsoft.Orleans.Core](https://www.nuget.org/packages/Microsoft.Orleans.Core)
     + [Microsoft.Orleans.CodeGenerator.MSBuild](https://www.nuget.org/packages/Microsoft.Orleans.CodeGenerator.MSBuild)
   * **RpcDemo.Hosting.Console**專案：Silo後端服務專案，安裝套件有：
     + [Microsoft.Orleans.Server](https://www.nuget.org/packages/Microsoft.Orleans.Server)
     + [Microsoft.Extensions.Logging.Console](https://www.nuget.org/packages/Microsoft.Extensions.Logging.Console)
     + [Microsoft.Extensions.Logging.Debug](https://www.nuget.org/packages/Microsoft.Extensions.Logging.Debug)
   * **RpcDemo.Client.Console**專案：示範呼叫Orleans RPC方法的Client專案，安裝套件有：
     + [Microsoft.Orleans.Client](https://www.nuget.org/packages/Microsoft.Orleans.Client)
2. 用Visual Studio Code開啟專案根目錄，如果有提示增加設定檔案：  
     
   選Yes，因為等下我們會修改它以便在Visual Studio Code內啟動除錯階段來跑。
3. 在**RpcDemo.Interfaces.Hello**專案內新增一個 IHelloGrain.cs 檔案，將Day03設計的RPC介面程式碼貼上：

* using System.Threading.Tasks;  
  using Orleans;  
    
  namespace RpcDemo.Interfaces.Hello;  
    
  public interface IHelloGrain : IGrainWithIntegerKey  
  {  
   Task<string> SayHello(string greeting);  
  }

1. 在**RpcDemo.Grains.Greeting**專案內新增一個 HelloGrain.cs 檔案，將Day03設計的Grain實作程式碼貼上：

* using System.Threading.Tasks;  
  using Orleans;  
  using RpcDemo.Interfaces.Hello;  
    
  namespace RpcDemo.Grains.Greeting;  
    
  public class HelloGrain : Grain, IHelloGrain  
  {  
   public Task<string> SayHello(string greeting)  
   {  
   return Task.FromResult($"Hello {greeting}!");  
   }  
  }

1. 將**RpcDemo.Hosting.Console** 專案中原本的 Program.cs 修改為：

* using System.Net;  
  using Microsoft.Extensions.Logging;  
  using Orleans;  
  using Orleans.Configuration;  
  using Orleans.Hosting;  
  using RpcDemo.Grains.Greeting;  
    
  var siloHost = new SiloHostBuilder()  
   .UseLocalhostClustering()  
   .Configure<ClusterOptions>(options =>  
   {  
   options.ClusterId = "console-host-01";  
   options.ServiceId = "Demo Greeting Service";  
   })  
   .Configure<EndpointOptions>(options => options.AdvertisedIPAddress = IPAddress.Loopback)  
   .ConfigureApplicationParts(parts =>  
   {  
   parts.AddApplicationPart(typeof(HelloGrain).Assembly).WithReferences();  
   })  
   .ConfigureLogging(logging =>  
   {  
   logging.AddConsole();  
   logging.AddDebug();  
   })  
   .Build();  
    
  //Tricks to manually wait for Ctrl+C key press  
  var waitForProcessShutdown = new ManualResetEvent(false);  
  Console.CancelKeyPress += (sender, eventArgs) =>  
  {  
   eventArgs.Cancel = true;  
   waitForProcessShutdown.Set();  
  };  
    
  await siloHost.StartAsync();  
  Console.WriteLine("===\r\nOrleans Silo started and able to connect,\r\nPress Ctrl+C to shutdown when client finish demonstration...\r\n===");  
  waitForProcessShutdown.WaitOne();  
    
  Console.WriteLine("Shutting down Silo...");  
  await siloHost.StopAsync().ConfigureAwait(false);  
  Console.WriteLine("===\r\nSilo shutdown complete, exiting...\r\n===");  
  Environment.Exit(0);

1. 將**RpcDemo.Client.Console**專案中原本的 Program.cs 修改為：

* using Orleans;  
  using Orleans.Configuration;  
  using RpcDemo.Interfaces.Hello;  
    
  using static System.Console;  
    
  WriteLine("\r\n---Orleans RPCDemo Client---");  
  WriteLine("\r\n---\r\nInitializing Orleans Client...\r\n---");  
  var client = new ClientBuilder()  
   .UseLocalhostClustering()  
   .Configure<ClusterOptions>(options =>  
   {  
   options.ClusterId = "console-client-01";  
   options.ServiceId = "Demo Greeting Service";  
   })  
   .ConfigureApplicationParts(parts =>  
   {  
   parts.AddApplicationPart(typeof(IHelloGrain).Assembly).WithReferences();  
   })  
   .Build();  
    
  WriteLine(  
   "Please wait until Orleans Server is started and ready for connections, then press any key to start connect...");  
  ReadKey();  
  await client.Connect();  
  WriteLine("\r\n---\r\nOrleans Client connected\r\n---");  
    
  var helloGrain = client.GetGrain<IHelloGrain>(0);  
  var helloResult = await helloGrain.SayHello("Orleans");  
  WriteLine($"\r\n---\r\nCall HelloGrain.SayHello(\"Orleans\") =\r\n{helloResult}\r\n---");  
  WriteLine("Demonstration finished, press any key to exit...");  
  ReadKey();  
    
  await client.Close();  
  client.Dispose();

1. 將Visual Studio Code的.vscode目錄內的task.json改為下列內容：

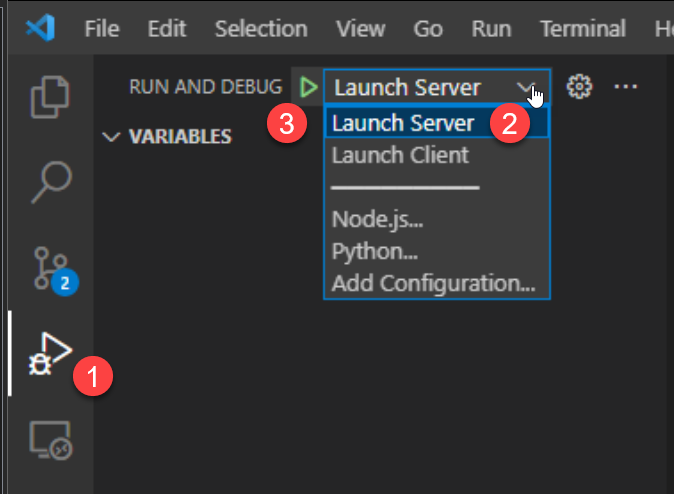
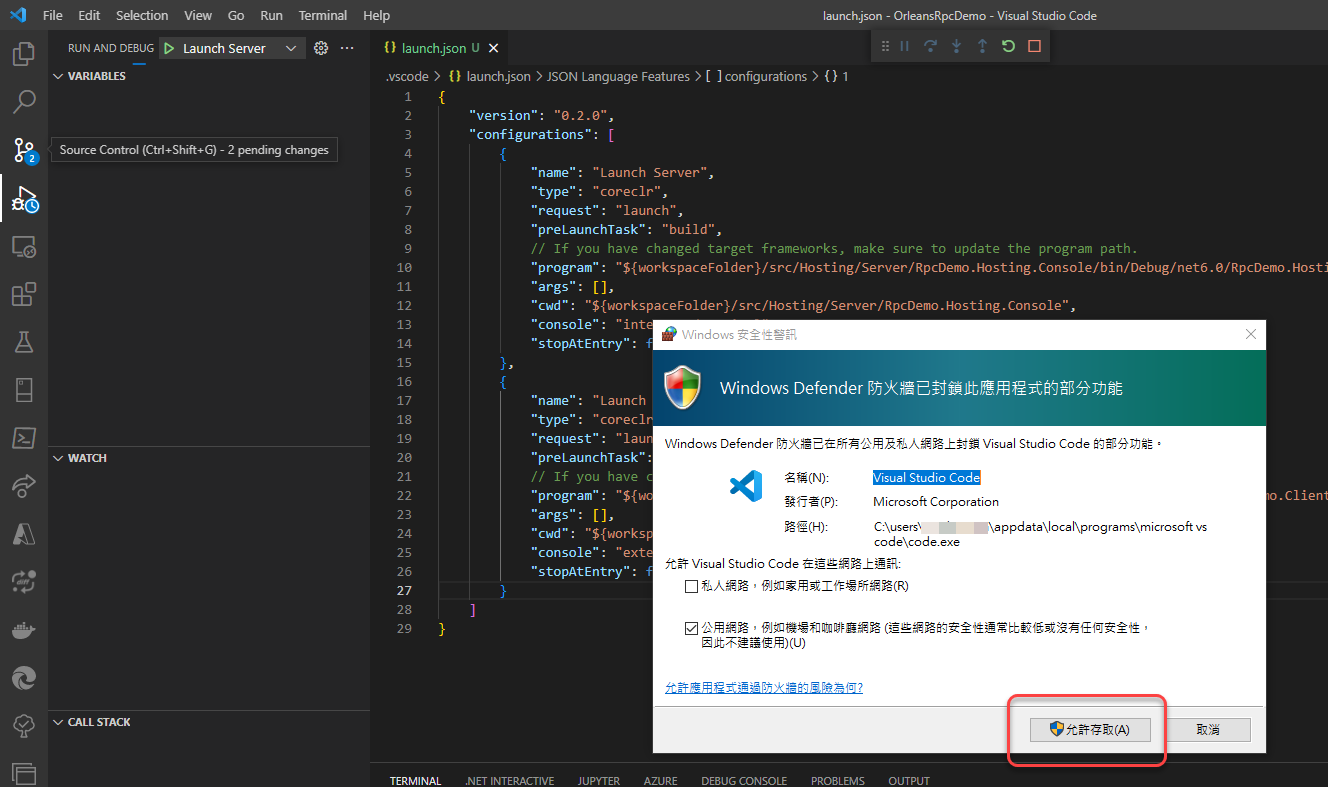
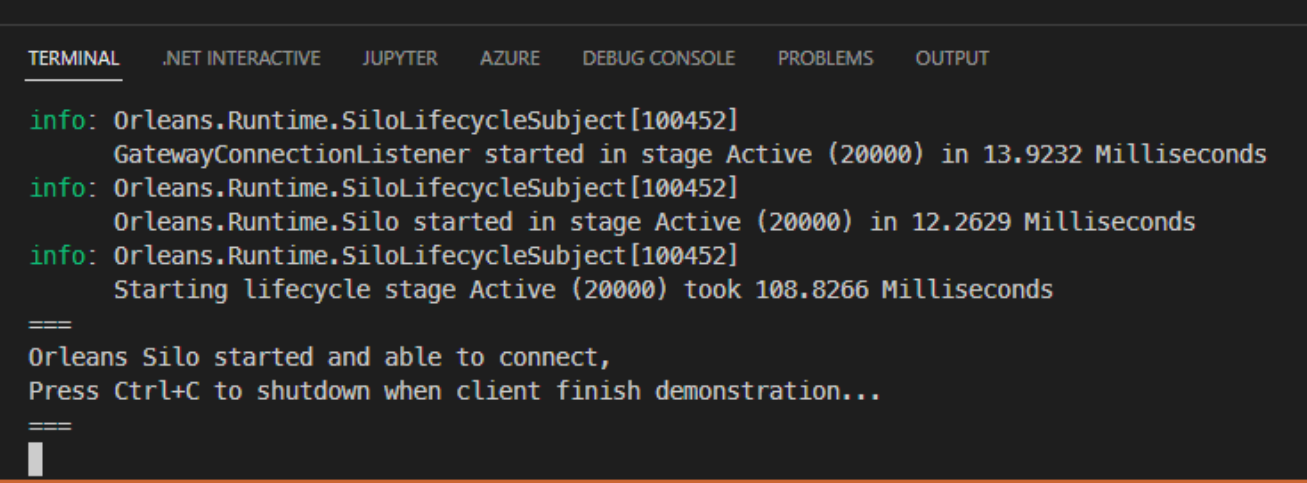
* {  
   "version": "2.0.0",  
   "tasks": [  
   {  
   "label": "build",  
   "dependsOn": [  
   "build server",  
   "build client"  
   ],  
   "dependsOrder": "sequence",  
   "group": "build"  
   },  
   {  
   "label": "build client",  
   "command": "dotnet",  
   "type": "process",  
   "args": [  
   "build",  
   "${workspaceFolder}/src/Hosting/Client/RpcDemo.Client.Console/RpcDemo.Client.Console.csproj",  
   "/property:GenerateFullPaths=true",  
   "/consoleloggerparameters:NoSummary"  
   ],  
   "problemMatcher": "$msCompile"  
   },  
   {  
   "label": "build server",  
   "command": "dotnet",  
   "type": "process",  
   "args": [  
   "build",  
   "${workspaceFolder}/src/Hosting/Server/RpcDemo.Hosting.Console/RpcDemo.Hosting.Console.csproj",  
   "/property:GenerateFullPaths=true",  
   "/consoleloggerparameters:NoSummary"  
   ],  
   "problemMatcher": "$msCompile"  
   }  
   ]  
  }

1. 將Visual Stuido Code的.vscode目錄內的launch.json改為下列內容：

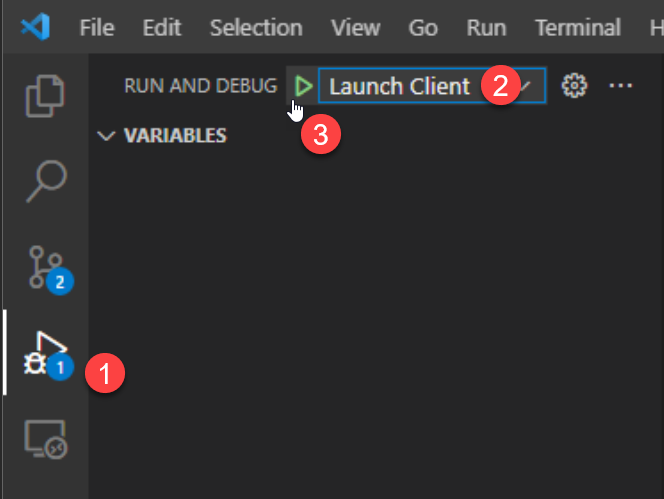
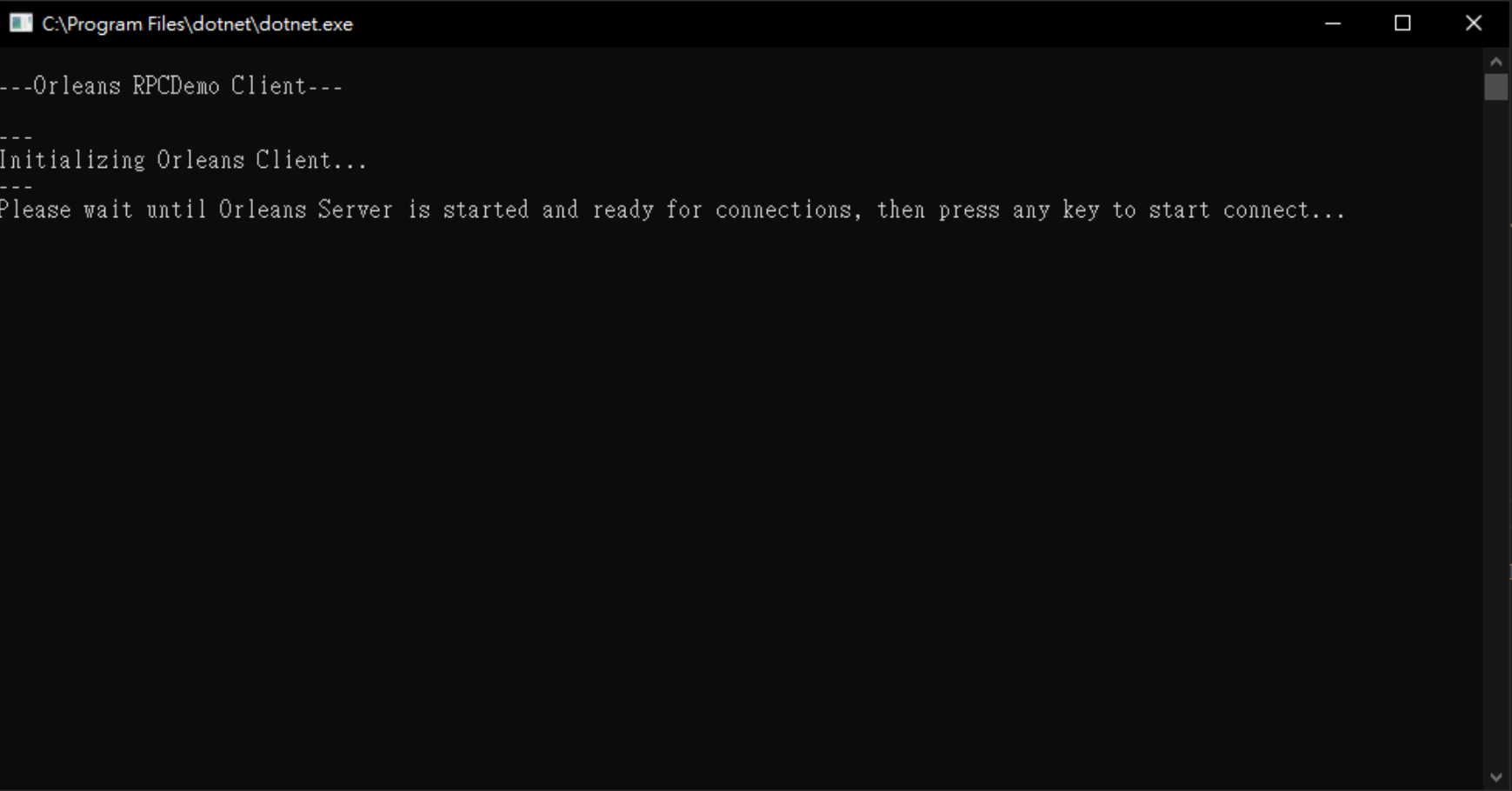
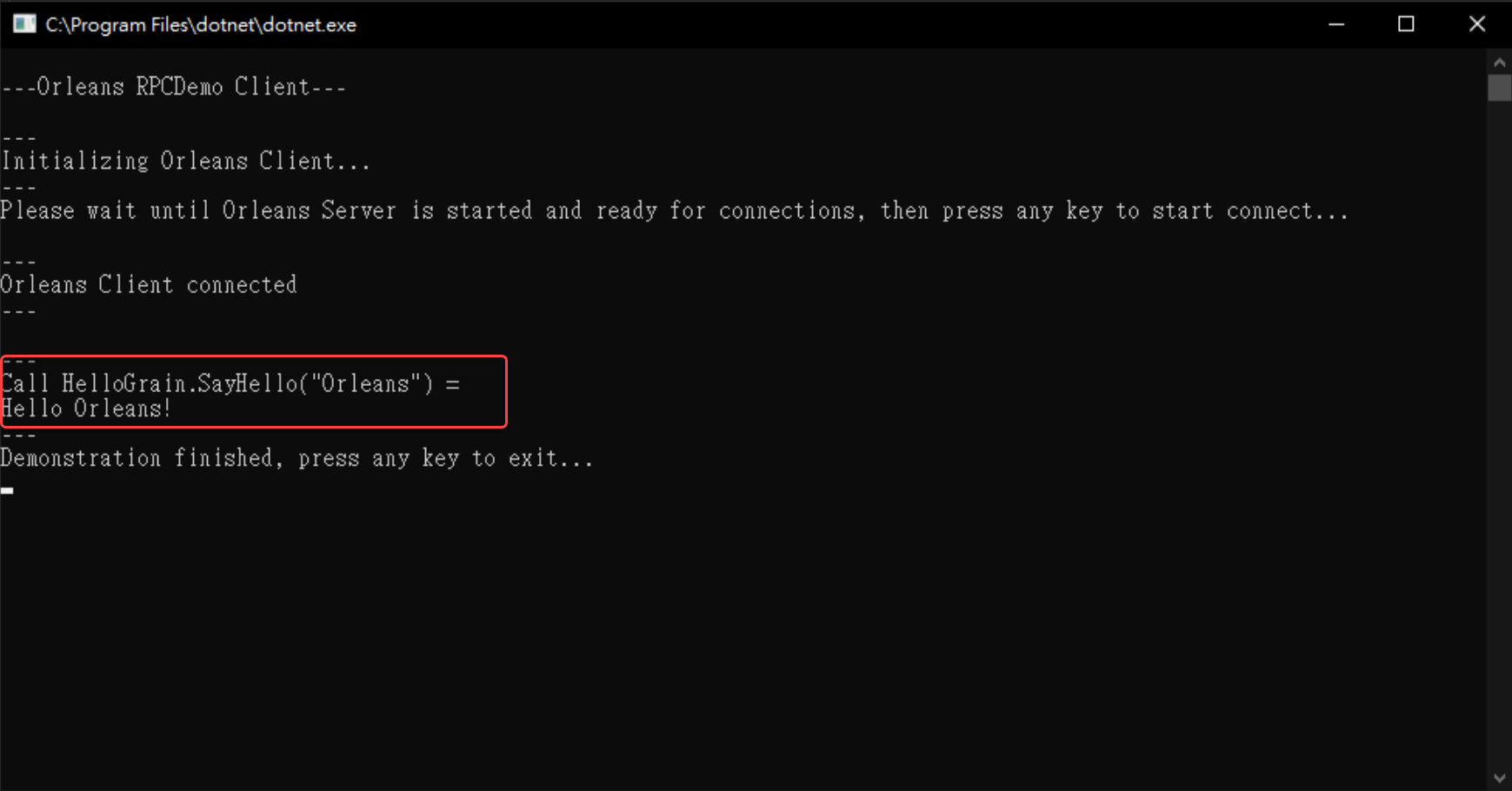
* {  
   "version": "0.2.0",  
   "configurations": [  
   {  
   "name": "Launch Server",  
   "type": "coreclr",  
   "request": "launch",  
   "preLaunchTask": "build server",  
   // If you have changed target frameworks, make sure to update the program path.  
   "program": "${workspaceFolder}/src/Hosting/Server/RpcDemo.Hosting.Console/bin/Debug/net6.0/RpcDemo.Hosting.Console.dll",  
   "args": [],  
   "cwd": "${workspaceFolder}/src/Hosting/Server/RpcDemo.Hosting.Console",  
   "console": "integratedTerminal",  
   "stopAtEntry": false  
   },  
   {  
   "name": "Launch Client",  
   "type": "coreclr",  
   "request": "launch",  
   "preLaunchTask": "build client",  
   // If you have changed target frameworks, make sure to update the program path.  
   "program": "${workspaceFolder}/src/Hosting/Client/RpcDemo.Client.Console/bin/Debug/net6.0/RpcDemo.Client.Console.dll",  
   "args": [],  
   "cwd": "${workspaceFolder}/src/Hosting/Client/RpcDemo.Client.Console",  
   "console": "externalTerminal",  
   "stopAtEntry": false  
   }  
   ]  
  }

## 執行測試

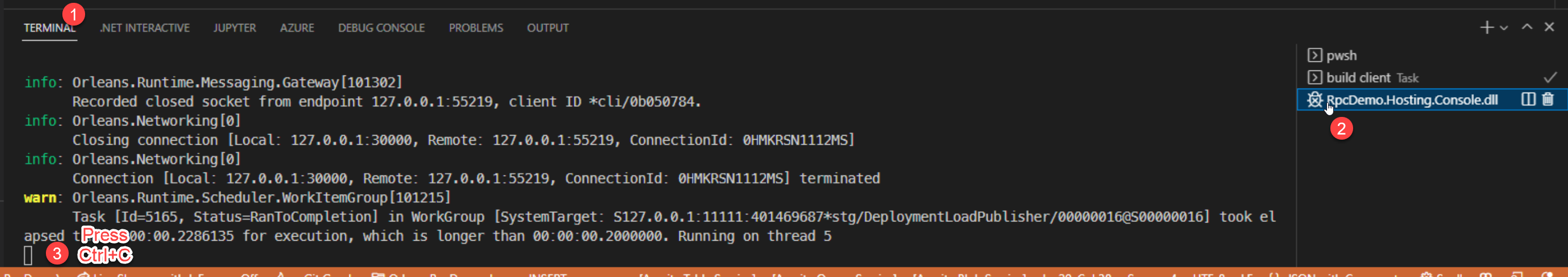
1. 切換到Visual Studio Code的**Debug**頁籤，選取**Launch Server**，按左邊的綠色啟動按鈕或下鍵盤**F5**鍵，啟動Orleans Silo。

* 
* 可能會出現是否要允許防火牆通過，請選擇**允許存取**。
* 
* 等到Visual Studio Code的Terminal視窗出現如下圖的螢幕提示時，表示Server端程式已經啟動完成。
* 

1. 切換到Visual Studio Code的**Debug**頁籤，選取**Launch Client**，按左邊的綠色啟動按鈕或下鍵盤**F5**鍵，啟動Orleans Client。

* 
* 在跳出來的命令列視窗中，按下任意鍵，讓Client端程式開始連線到Server端程式。
* 
* 然後就會顯示呼叫Grain的SayHello() RPC方法的結果：
* 

1. 要結束此執行範例，在Client的命令列視窗按任意鍵停止Client端程式，然後在Visual Studio的Terminal視窗按下**Ctrl+C**停止Server端程式。

* 

整個完成的範例程式GitHub專案在：<https://github.com/windperson/OrleansRpcDemo/tree/day06>

明天將會介紹另一種使用[.NET Interactive Notebooks](https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.dotnet-interactive-vscode)來驗證Grain RPC方法呼叫的機制，比較不需要準備這麼多程式碼專案。