

РaaS - Розгортання веб-сервісів WCF C Sharp за допомогою Microsoft Azure App Service

Мета: Отримати практичний досвід розробки веб-сервісів C# WCF та їх розгортання на хмарній РaaS-платформі Microsoft Azure App Service, а також створення та запуск клієнтів веб-сервісів.

Завдання:

1. Дослідити веб-сервіс C# WCF.
2. Протестувати веб-сервіс локально (WCF Test Client / SoapUI).
3. Дослідити клієнт C# для веб-сервісу.
4. Розгорнути веб-сервіс WCF в Azure App Service.
5. Налаштувати клієнт на роботу з хмарним сервісом.

Лабораторне середовище:

- Visual Studio (workload "Azure development").
- Microsoft Azure Subscription (Free Tier / Student).

Крок 0: Завантаження проекту

Використовуйте той самий архів `repos.zip`, що був наданий до оригінальної лабораторної роботи. Розпакуйте його у зручну папку (наприклад, `C:\Users\ВашеІм'я\source\repos`).

Крок 1: Запуск та дослідження локально

Оскільки проект розділений на два окремих рішення (Server і Client), нам потрібно запустити їх паралельно у двох різних вікнах Visual Studio.

1.1. Запуск Сервера (WCF Service)

1. Зайдіть у папку `WcfServiceCalc`.
2. Відкрийте файл `WcfServiceCalc.sln`.
3. Якщо Visual Studio запропонує оновити версію .NET Framework — погодьтеся.
4. Натисніть **F5** (або кнопку "Start"), щоб запустити сервіс у режимі налагодження.

5. Відкриється браузер або **WCF Test Client**.
6. **ВАЖЛИВО:** Не закривайте це вікно Visual Studio і не зупиняйте налагодження. Сервер має працювати, щоб клієнт міг до нього підключитися.

localhost - /

28.07.2020	17:42	<dir> App_Data
24.11.2025	20:07	<dir> bin
01.09.2020	22:07	255 default.html
31.08.2020	18:49	744 IService1.cs
30.08.2020	23:19	<dir> obj
30.08.2020	23:19	<dir> Properties
28.07.2020	17:42	111 Service1.svc
28.07.2020	17:46	1002 Service1.svc.cs
01.09.2020	22:34	5315 WcfServiceCalc.csproj
01.09.2020	22:34	1457 WcfServiceCalc.csproj.user
31.08.2020	1:15	1141 WcfServiceCalc.sln
28.07.2020	17:42	1156 Web.config
28.07.2020	17:42	1300 Web.Debug.config
28.07.2020	17:42	1361 Web.Release.config

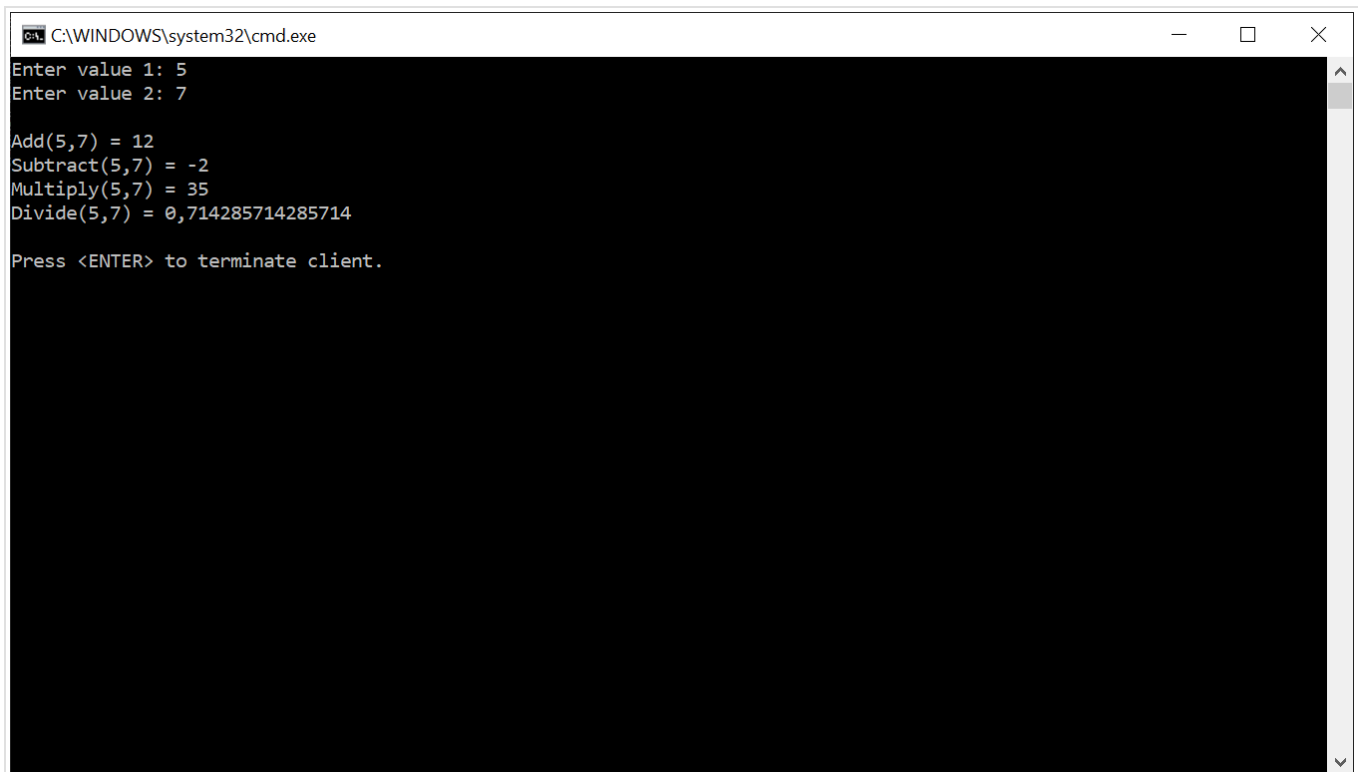
Місце для скріншота: Запущений WCF Service (браузер або Test Client)

1.2. Запуск Клієнта (WCF Client)

1. Відкрийте **нове, друге вікно** Visual Studio (не закриваючи перше).
2. Через меню *File -> Open -> Project/Solution* знайдіть папку `WcfClientCalc`.
3. Відкрийте файл `WcfClientCalc.sln`.
4. Так само погодьтеся на оновлення .NET Framework, якщо запропонують.
5. Натисніть **F5** для запуску клієнта.

1.3. Перевірка зв'язку

1. Після запуску клієнта відкриється **консольне вікно**.
2. Введіть два числа за запитом програми (наприклад, 5 і 7).
3. Програма надішле запит на ваш локальний сервер (який ви запустили у пункті 1.1) і виведе результати операцій (Add, Subtract тощо).
4. Якщо ви бачите розраховані результати — локальне середовище налаштовано вірно.



```
C:\WINDOWS\system32\cmd.exe
Enter value 1: 5
Enter value 2: 7

Add(5,7) = 12
Subtract(5,7) = -2
Multiply(5,7) = 35
Divide(5,7) = 0,714285714285714

Press <ENTER> to terminate client.
```

Місце для скріншота: Консоль клієнта з результатами обчислень

Крок 2: Публікація веб-сервісу в Azure App Service

Ми будемо використовувати **Azure App Service (Windows)**, оскільки WCF нативно працює на Windows.

0. Налаштування Web.config для Azure (HTTPS). Оскільки Azure використовує захищене з'єднання (HTTPS), нам потрібно налаштувати WCF сервіс для роботи з ним.
 1. У проєкті `WcfServiceCalc` відкрийте файл `Web.config`.
 2. Знайдіть секцію `<system.serviceModel>`.
 3. Замініть **весь вміст** цієї секції на наступний код:

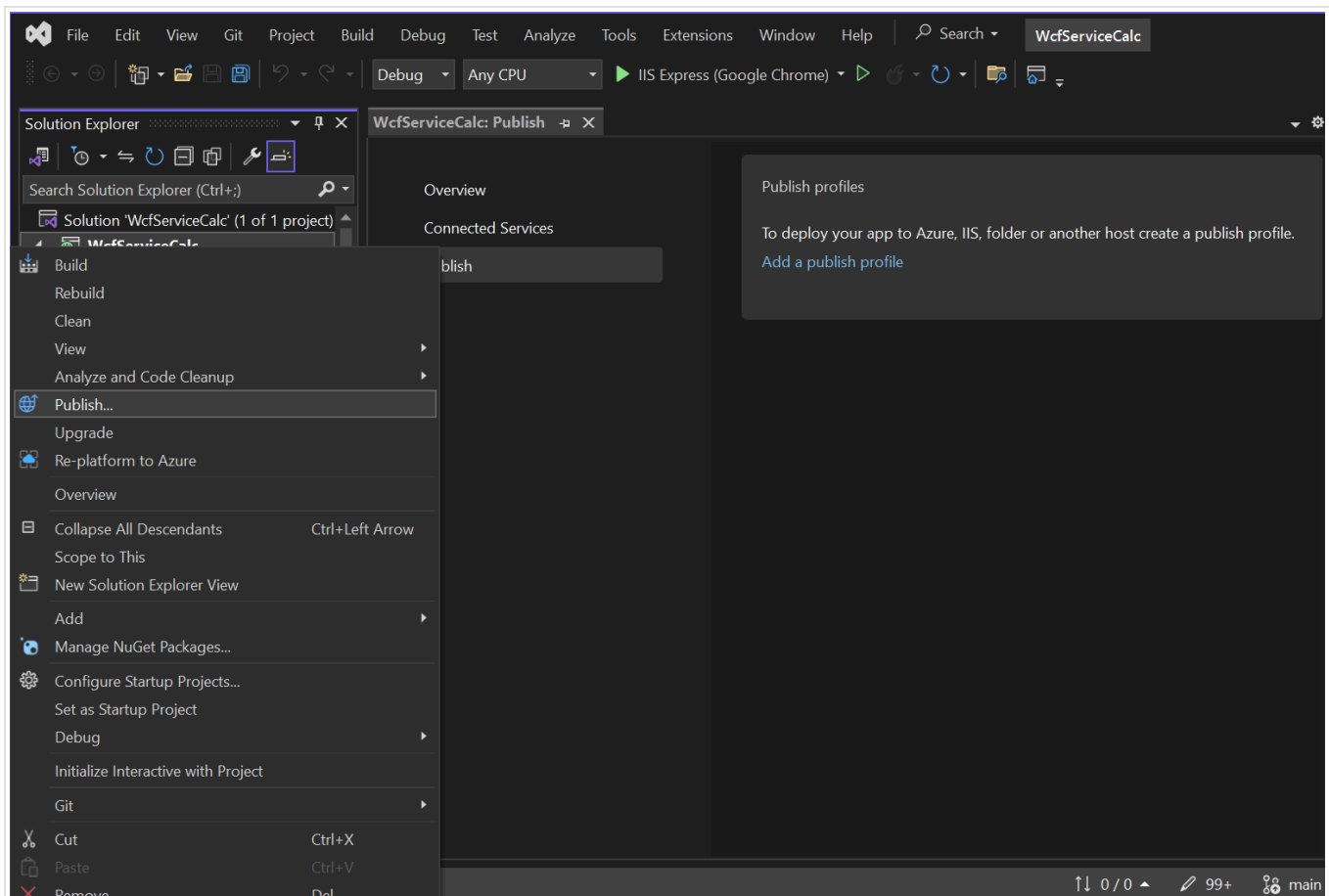
```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="SecureBinding">
        <security mode="Transport">
          <transport clientCredentialType="None"/>
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
```

```

<protocolMapping>
  <add binding="basicHttpBinding" scheme="https"
bindingConfiguration="SecureBinding" />
</protocolMapping>
<behaviors>
  <serviceBehaviors>
    <behavior>
      <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
      <serviceDebug includeExceptionDetailInFaults="false"/>
    </behavior>
  </serviceBehaviors>
</behaviors>
<serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
</system.serviceModel>

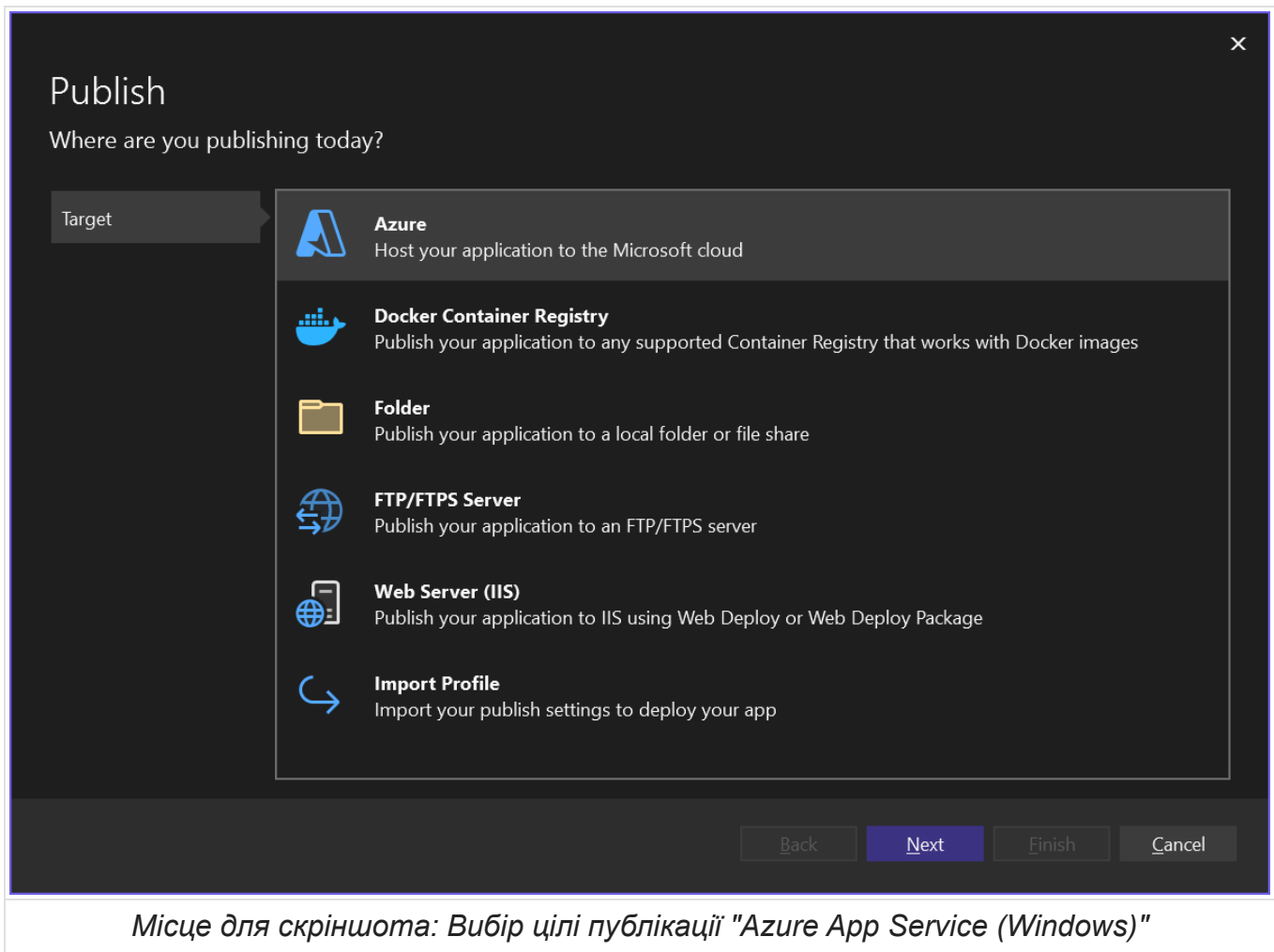
```

1. У Solution Explorer натисніть правою кнопкою миші на проект **WcfServiceCalc** (Сервер) і оберіть **"Publish..."** (Опублікувати).

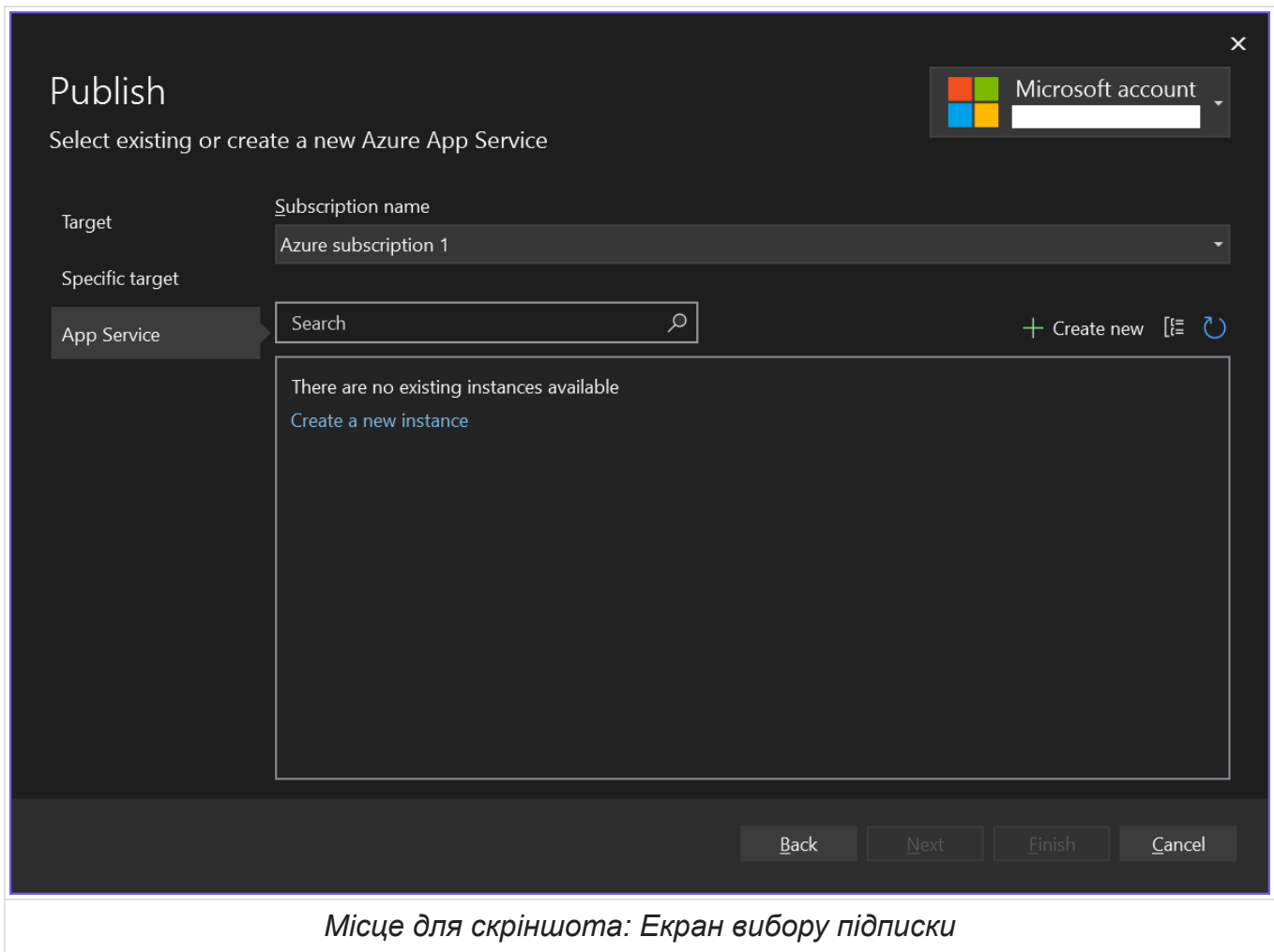


Місце для скріншота: Контекстне меню проекту з виділеним пунктом Publish

2. У вікні майстра публікації оберіть **Azure** і натисніть "Next".
3. Оберіть **Azure App Service (Windows)** і натисніть "Next".



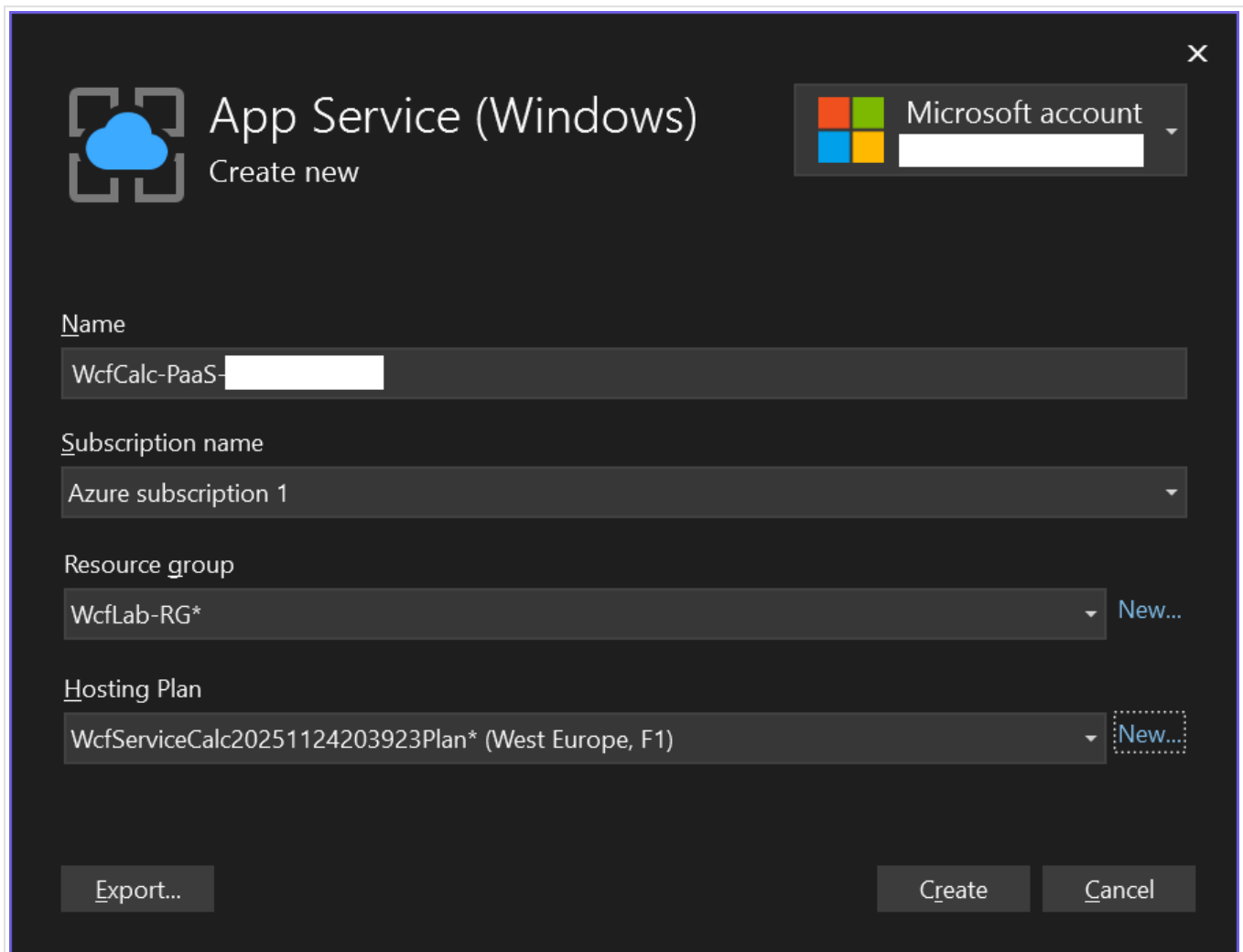
- У наступному вікні натисніть **Specific target -> App Service (Windows)**, **Next** та **Finish**, щоб створити новий ресурс.
- Якщо у вас не було завантажено необхідне доповнення, вискочить **VS Installer** та запропонує довантажити його. Погодьтеся (та за необхідності повторіть кроки 1-3).
- Увійдіть в акаунт **Microsoft**, у якому ви активували підписку.
 - * **Size**: Оберіть **Free (F1)** або **Shared (D1)**, щоб уникнути витрат.



8. Натисніть кнопку **Create new +**.

9. Заповніть поля діалогового вікна "App Service (Windows)":

- **Name:** Введіть унікальне ім'я (наприклад, `WcfCalc-PaaS-[ВашеПрізвище]`). Це ім'я стане частиною URL-адреси (`.azurewebsites.net`).
- **Subscription:** Оберіть вашу активну підписку.
- **Resource Group:** Натисніть "New" і створіть групу, наприклад, `WcfLab-RG` .
- **Hosting Plan:** Створіть новий план.
 - **Location:** Оберіть найближчий регіон (наприклад, `West Europe`).
 - **Size:** Оберіть **Free (F1)** або **Shared (D1)**, щоб уникнути витрат.



App Service (Windows)
Create new

Microsoft account

Name
WcfCalc-PaaS-

Subscription name
Azure subscription 1

Resource group
WcfLab-RG* New...

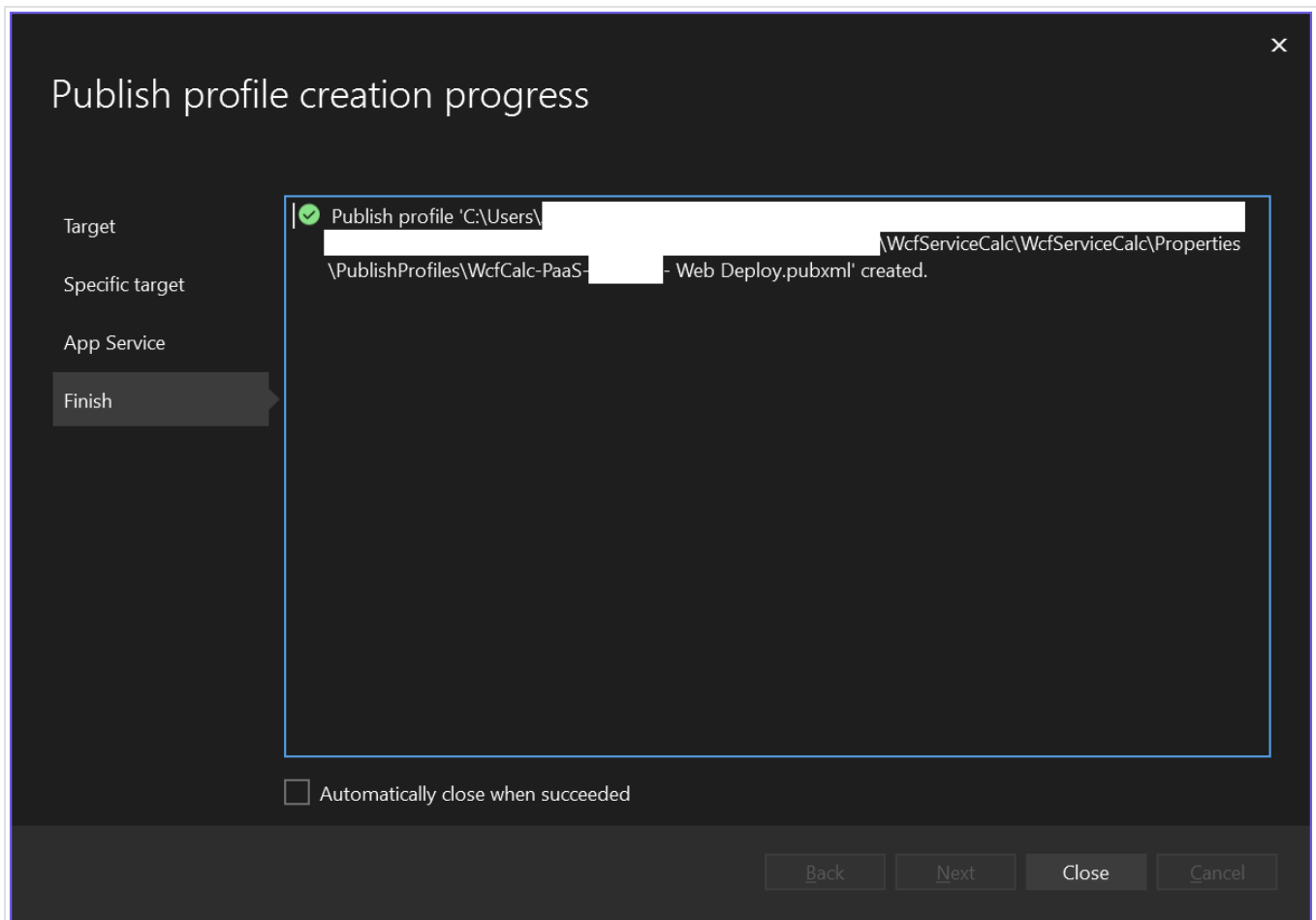
Hosting Plan
WcfServiceCalc20251124203923Plan* (West Europe, F1) New...

Export... Create Cancel

Місце для скріншота: Налаштування створення App Service (вказання імені та плану)

9. Натисніть **Create**. Зачекайте, поки ресурси створюються в Azure.
10. Після створення натисніть **Finish**.

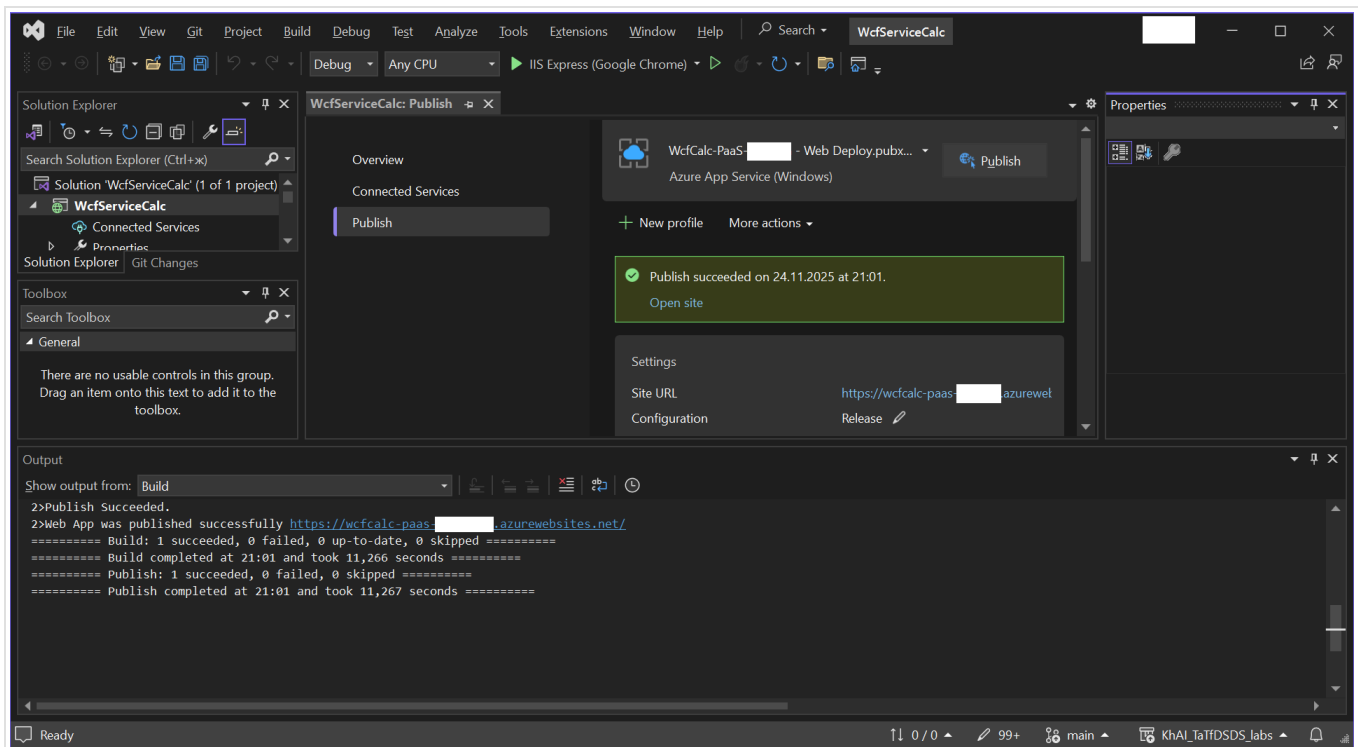
* *Примітка:* Якщо ви побачите помилку 403/404 або "Directory Listing Denied" – це нормально, бо за кореневою адресою `/` нічого немає. Головне, що сервіс працює за своєю адресою.



Місце для скріншота: Вікно Visual Studio з повідомленням "Publish profile: Succeeded"

11. Ви повернетесь на головний екран публікації. Натисніть кнопку **Publish** у верхньому правому куті.
12. Слідкуйте за вікном **Output**. Коли публікація завершиться, Visual Studio автоматично відкриє ваш браузер.

* *Примітка:* Якщо ви побачите помилку 403/404 або "Directory Listing Denied" – це нормально, бо за кореневою адресою `/'` нічого немає. Головне, що сервіс працює за своєю адресою.



Місце для скріншота: Вікно Output у Visual Studio з повідомленням "Publish: Succeeded"

Крок 3: Перевірка розгорнутого сервісу

1. У браузері, що відкрився, допишіть до адреси: `/Service1.svc`.
 - Повна адреса має виглядати так: `https://wcfcalc-paas-[ВашеПрізвище].azurewebsites.net/Service1.svc`
2. Ви повинні побачити службову сторінку "Service1 Service" з повідомленням "You have created a service".

Service1 Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://wcfcalc-paas-siroklyn.azurewebsites.net/Service1.svc?wsdl
```

You can also access the service description as a single file:

```
http://wcfcalc-paas-siroklyn.azurewebsites.net/Service1.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        Service1Client client = new Service1Client();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Visual Basic

Місце для скріншота: Браузер із відкритою сторінкою .svc в Azure

- Допишіть `?wsdl` в кінці URL (`https://.../Service1.svc?wsdl`), щоб переконатися, що опис сервісу доступний. Скопіюйте це повне посилання на WSDL (для кроку 4). повідомленням "You have created a service".

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<wsc:definitions xmlns:wsc="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:tns="http://tempuri.org/"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
  xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
  xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsa10="http://www.w3.org/2005/08/addressing" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" name="Service1"
  targetNamespace="http://tempuri.org/">
  <wsc:types>
    <xsd:schema targetNamespace="http://tempuri.org/Imports">
      <xsd:import schemaLocation="https://wcfcalc-paas-[redacted].azurewebsites.net/Service1.svc?xsd=xsd0" namespace="http://tempuri.org/" />
      <xsd:import schemaLocation="https://wcfcalc-paas-[redacted].azurewebsites.net/Service1.svc?xsd=xsd1"
        namespace="http://schemas.microsoft.com/2003/10/Serialization/" />
    </xsd:schema>
  </wsc:types>
  <wsc:message name="IService1_Add_InputMessage">
    <wsc:part name="parameters" element="tns:Add"/>
  </wsc:message>
  <wsc:message name="IService1_Add_OutputMessage">
    <wsc:part name="parameters" element="tns:AddResponse"/>
  </wsc:message>
  <wsc:message name="IService1_Subtract_InputMessage">
    <wsc:part name="parameters" element="tns:Subtract"/>
  </wsc:message>
  <wsc:message name="IService1_Subtract_OutputMessage">
    <wsc:part name="parameters" element="tns:SubtractResponse"/>
  </wsc:message>
  <wsc:message name="IService1_Multiply_InputMessage">
    <wsc:part name="parameters" element="tns:Multiply"/>
  </wsc:message>
  <wsc:message name="IService1_Multiply_OutputMessage">
    <wsc:part name="parameters" element="tns:MultiplyResponse"/>
  </wsc:message>
```

Місце для скріншота: Браузер із відкритою сторінкою .svc?wsdl в Azure

- Якщо у вас, замість .xml файлу, видає той самий результат, що в пункті 3.2, виконайте наступні кроки:

Azure працює через захищений протокол HTTPS, але за замовчуванням старі проекти WCF дозволяють віддавати метадані тільки через HTTP. Щоб виправити це:

- Поверніться у **Visual Studio**.

2. У **Solution Explorer** розгорніть проект `WcfServiceCalc` і відкрийте файл `Web.config`.

3. Знайдіть секцію `<serviceBehaviors>` -> `<behavior>`.

4. Знайдіть рядок:

```
<serviceMetadata httpGetEnabled="true"/>
```

5. Змініть його, додавши дозвіл для HTTPS (`httpsGetEnabled="true"`):

```
<serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
```

6. Збережіть файл (**Ctrl+S**).

7. Знову опублікуйте проект в Azure: натисніть правою кнопкою миші на проект `WcfServiceCalc` -> **Publish...** -> кнопка **Publish**.

8. Після успішної публікації спробуйте знову відкрити посилання з `?wsdl` (наприклад, `https://.../Service1.svc?wsdl`). Тепер ви маєте побачити XML-код.

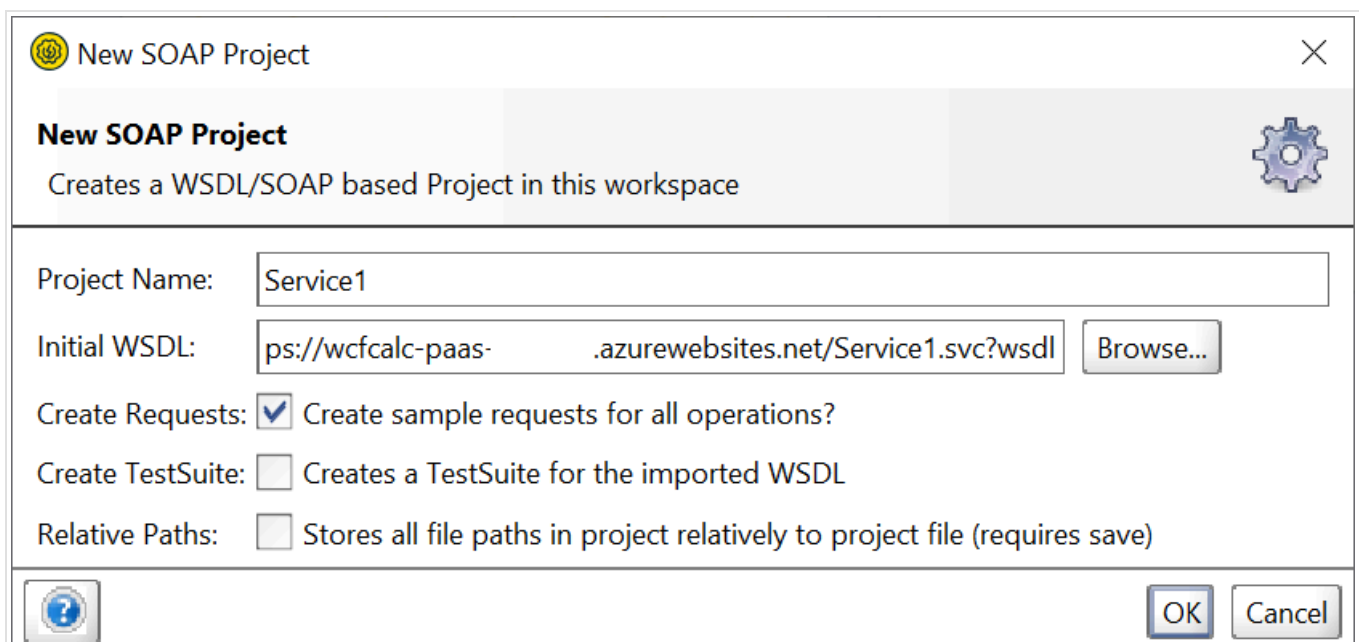
Крок 4: Тестування за допомогою SoapUI (Опціонально)

1. Відкрийте **SoapUI**.

2. Натисніть **"SOAP"** (або `File -> New SOAP Project`).

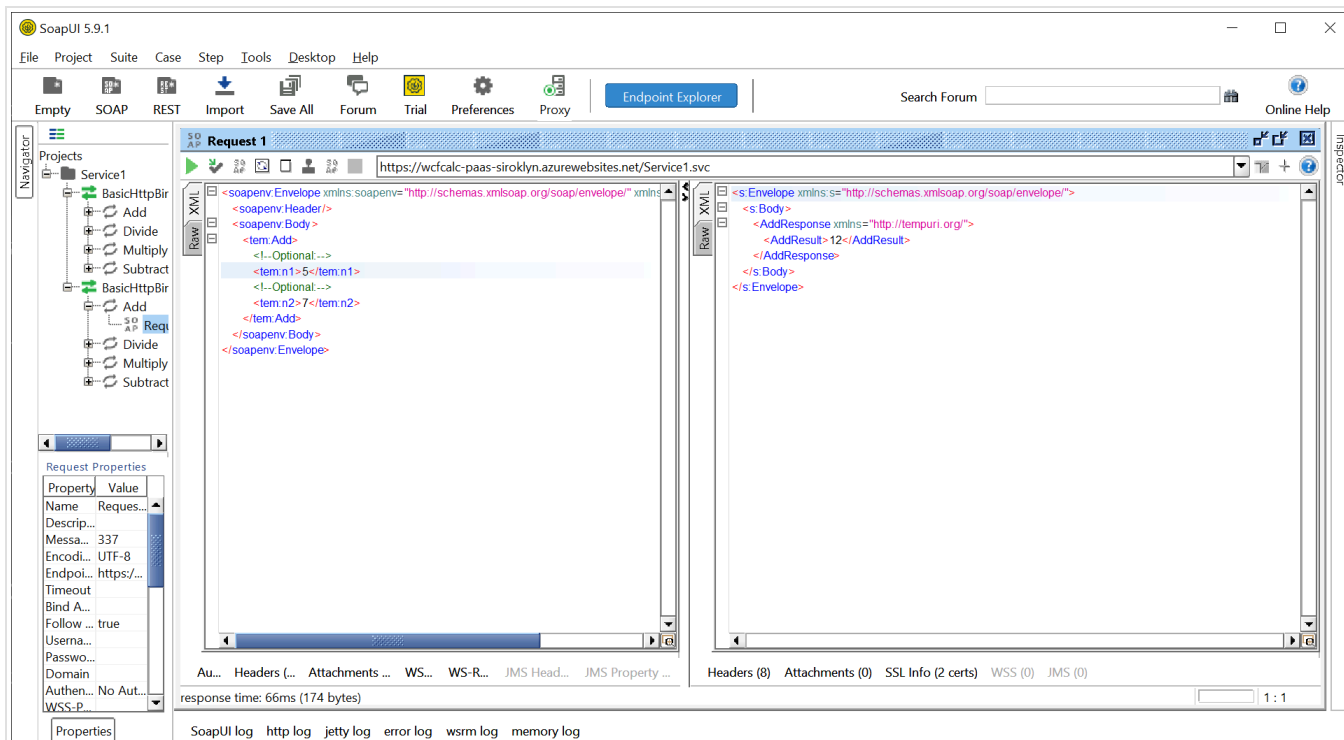
3. У полі **Initial WSDL** вставте URL-адресу вашого WSDL з Azure (з Кроку 3).

4. Натисніть **OK**. SoapUI завантажить визначення сервісу.
результатом додавання.



Місце для скріншота: Створення нового SOAP проекту

5. Розгорніть операцію Add , відкрийте Request 1 , введіть значення для n1 та n2 і натисніть зелену стрілку (Play).
6. Переконайтеся, що у правій частині вікна прийшла відповідь з результатом додавання.



Місце для скріншота: Успішний запит у SoapUI до хмарного сервісу

Крок 5: Налаштування C# Клієнта

Тепер налаштуємо консольну програму `WcfClientCalc` так, щоб вона зверталася до вашого сервісу в хмарі, а не локально.

1. У Solution Explorer розгорніть проект `WcfClientCalc` .
2. Відкрийте файл `app.config` .
3. Знайдіть секцію `<system.serviceModel>` .
4. Замініть її на цей код, не забудьте вставити вашу URL-адресу:

Було:

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IService1" />
    </basicHttpBinding>
  </bindings>
</system.serviceModel>
```

```

    </bindings>
    <client>
      <endpoint address="http://localhost:52788/Service1.svc"
binding="basicHttpBinding"
      bindingConfiguration="BasicHttpBinding_IService1"
contract="ServiceReference1.IService1"
      name="BasicHttpBinding_IService1" />
    </client>
  </system.serviceModel>

```

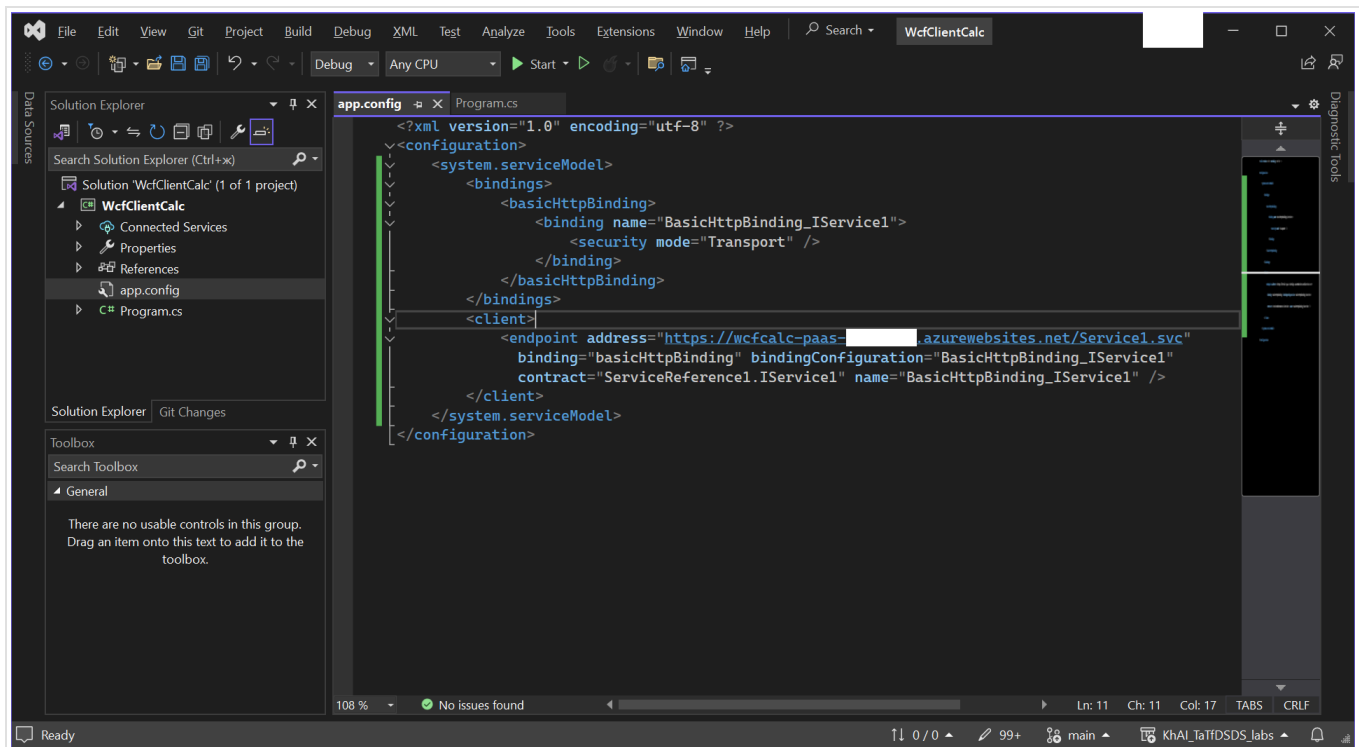
Має стати (приклад):

```

<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IService1">
        <security mode="Transport" />
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="https://wcfcalc-paas-
[ВашеПрізвище].azurewebsites.net/Service1.svc" binding="basicHttpBinding"
bindingConfiguration="BasicHttpBinding_IService1"
contract="ServiceReference1.IService1" name="BasicHttpBinding_IService1" />
  </client>
</system.serviceModel>

```

5. Збережіть файл (Ctrl+S).

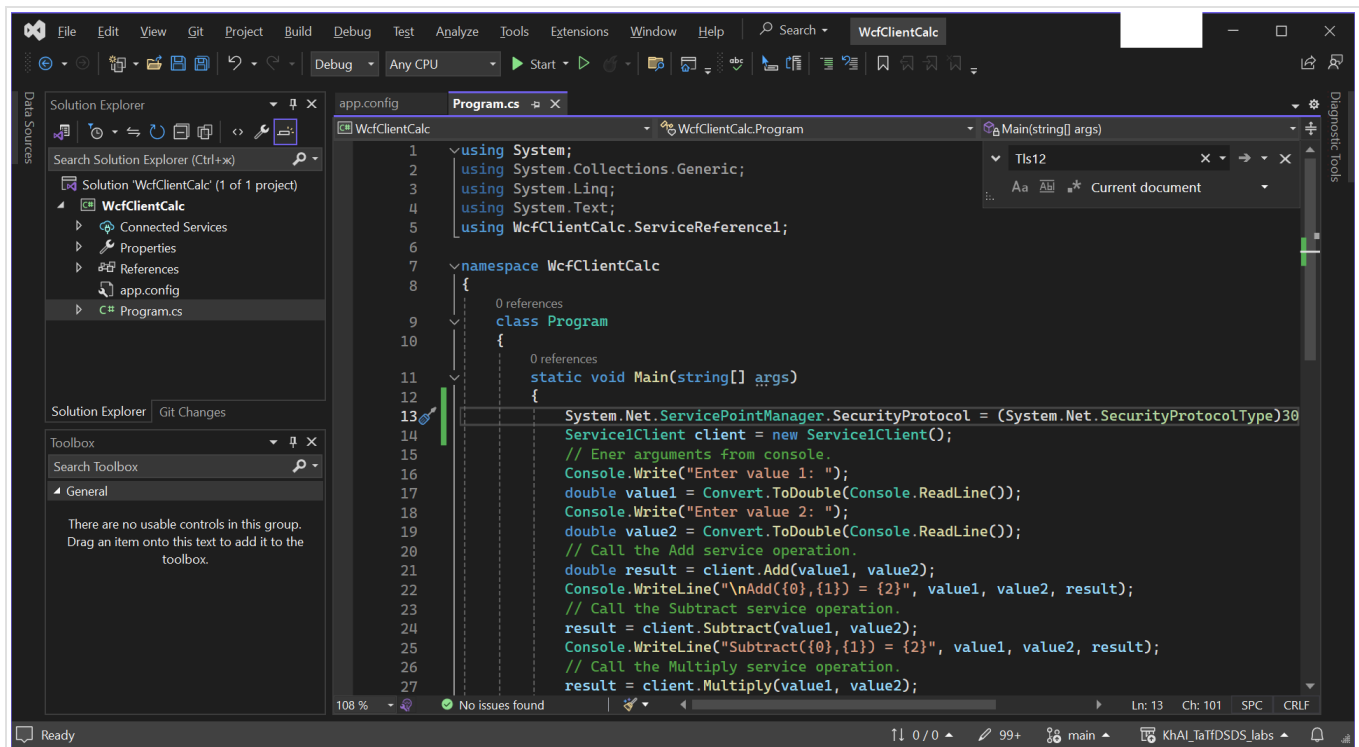


Місце для скріншота: Змінений файл app.config з новою URL-адресою

Старі версії .NET не підтримують TLS 1.2 за замовчуванням, що викликає помилку з'єднання з Azure.

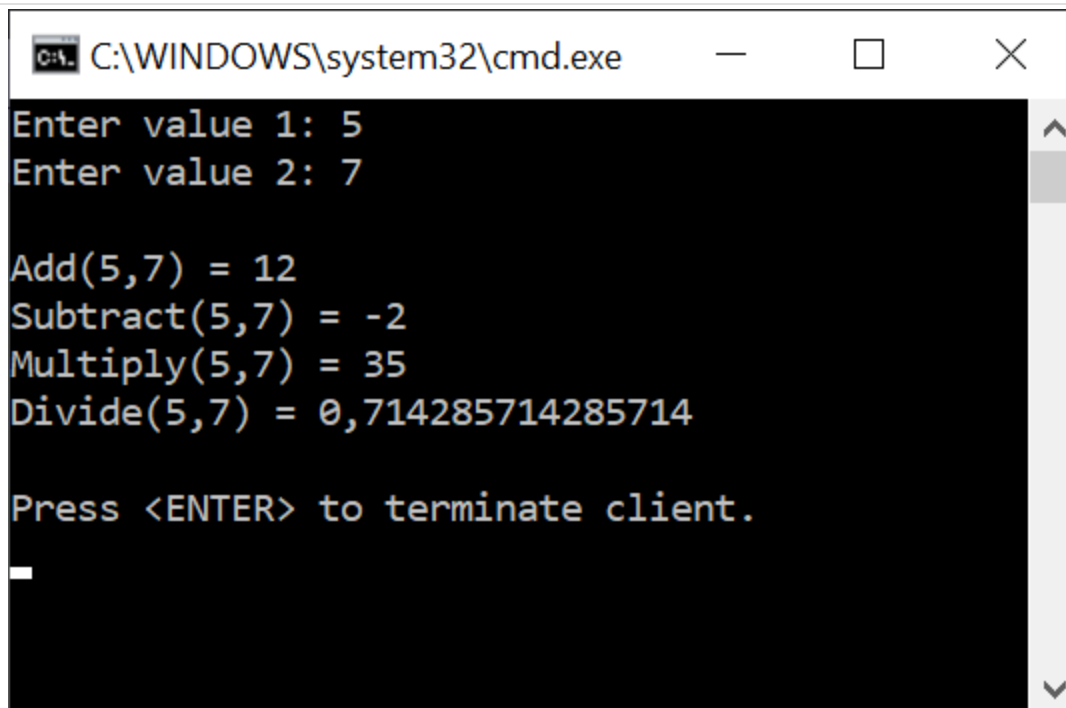
6. Відкрийте файл `Program.cs` у проекті `WcfClientCalc`.
7. Знайдіть метод `static void Main(string[] args)`.
8. Додайте цей рядок на самий початок методу:

```
// ПРИМУСОВЕ ВКЛЮЧЕННЯ TLS 1.2 для сумісності з Azure
System.Net.ServicePointManager.SecurityProtocol =
(System.Net.SecurityProtocolType)3072;
```



Місце для скріншота: Змінений файл Program.cs з новим рядком

9. Збережіть файл (Ctrl+S).
10. Натисніть **F5** для запуску клієнта.
11. Введіть числа у консолі. Якщо програма видає коректний результат, це означає, що ваш локальний клієнт успішно з'єднався з віддаленим сервером в Azure.

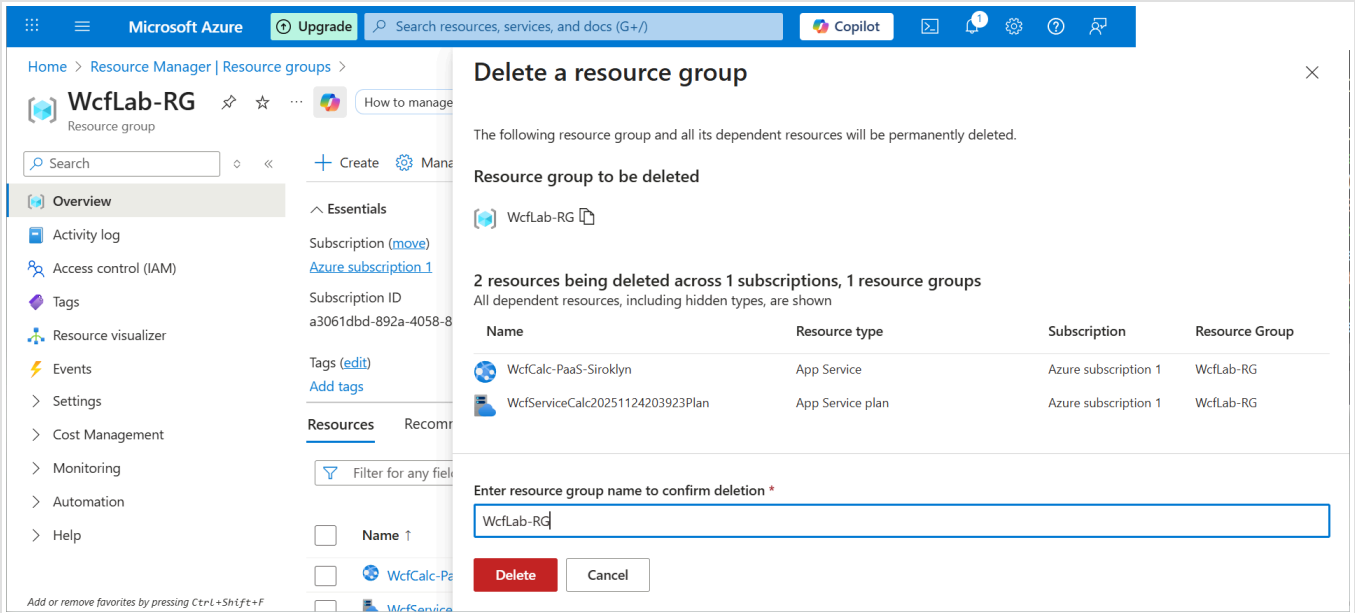


Місце для скріншота: Робота консольного клієнта, що отримує дані з Azure

Крок 6: Очищення ресурсів

ВАЖЛИВО: Після завершення роботи видаліть створені ресурси, щоб уникнути списання коштів з вашого рахунку Azure.

- 1. Зайдіть на портал portal.azure.com.
- 2. Перейдіть у розділ **"Resource groups"** (Групи ресурсів).
- 3. Знайдіть групу WcfLab-RG (яку ви створили на кроці 2).
- 4. Натисніть **"Delete resource group"**.
- 5. Введіть назву групи для підтвердження та натисніть **"Delete"**. Це видалить App Service та App Service Plan.



Місце для скріншота: Підтвердження видалення групи ресурсів