FaaS - Розробка безсерверного веб-сервісу з Azure Functions

Мета: Отримати практичний досвід у розробці та розгортанні безсерверних функцій у хмарі (FaaS) за допомогою **Azure Functions** та **Azure API Management**.

Завдання:

1. Створення **Azure Function App** (контейнера для функцій).
2. Створення функції-калькулятора (HTTP Trigger).
3. Тестування функції-калькулятора безпосередньо.
4. Створення **Azure API Management** (шлюзу API).
5. Створення методу `GET` для виклику функції (з перетворенням запиту).
6. Розгортання та тестування API.
7. Видалення ресурсів.

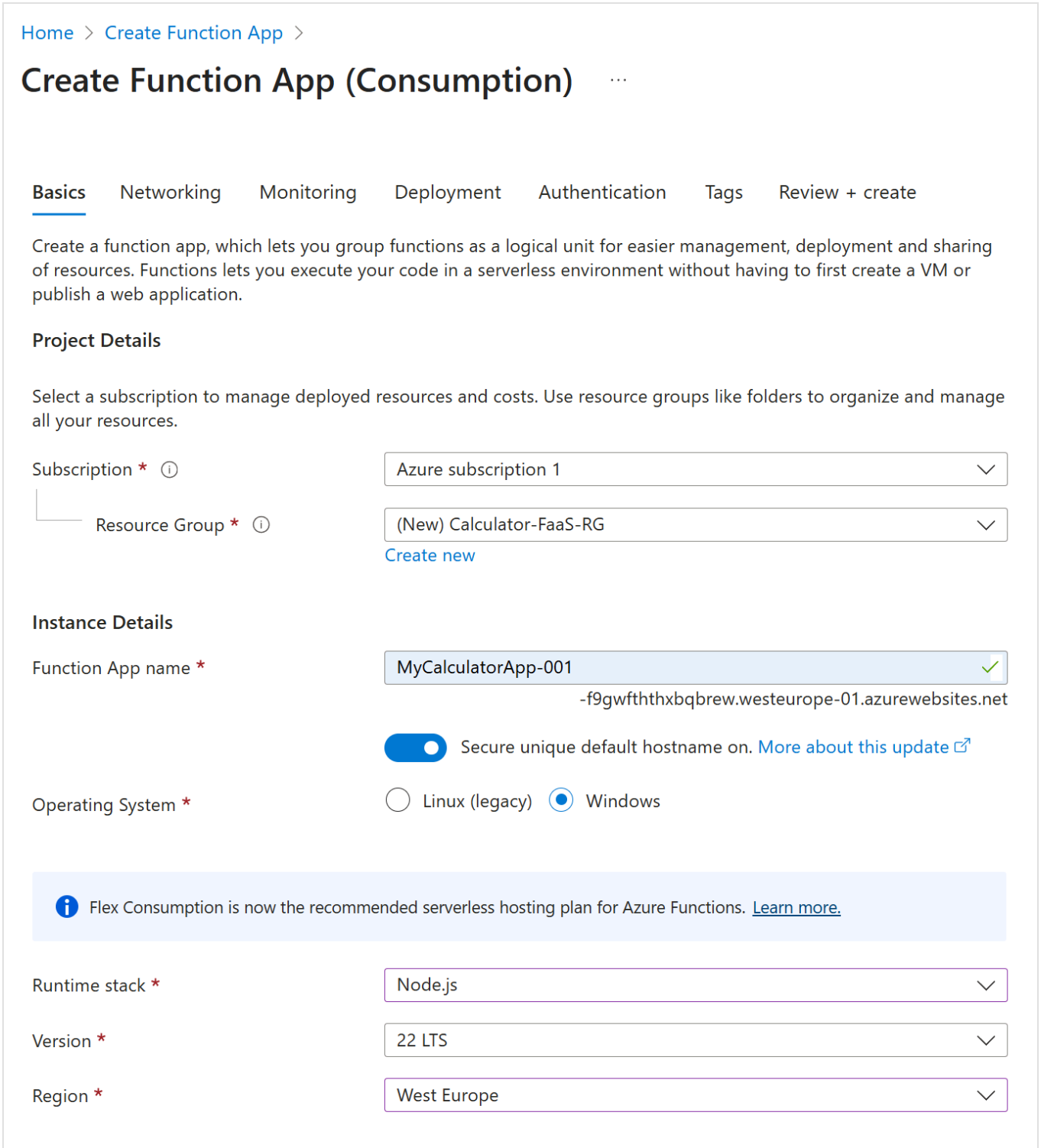
Лабораторне середовище:

- **SoapUI** (або Postman, або звичайний веб-браузер).

Крок 1: Створення Azure Function (Функції-калькулятора)

Спочатку ми створимо саму безсерверну функцію. В Azure це складається з двох частин: **Function App** (контейнер) та **Function** (код).

1. Увійдіть до порталу Azure: https://portal.azure.com/
2. Натисніть **"Create a resource"** (Створити ресурс) -> **"Function App"**.
3. На вкладці **"Basics"** заповніть поля:
   - **Hosting option:** Оберіть `Consumption (Serverless)`. Це аналог моделі оплати AWS Lambda.
   - **Resource Group:** Створіть нову, наприклад, `Calculator-FaaS-RG`.
   - **Function App name:** Введіть унікальне ім'я, наприклад, `MyCalculatorApp-` (додайте унікальні цифри).
   - **Operating System:** `Windows` (або `Linux`, це не критично для Node.js).
   - **Runtime stack:** Оберіть `Node.js` (наприклад, `Node.js 22 LTS`).
4. Натисніть **"Review + create"**, а потім **"Create"**. Розгортання займе 1-2 хвилини.

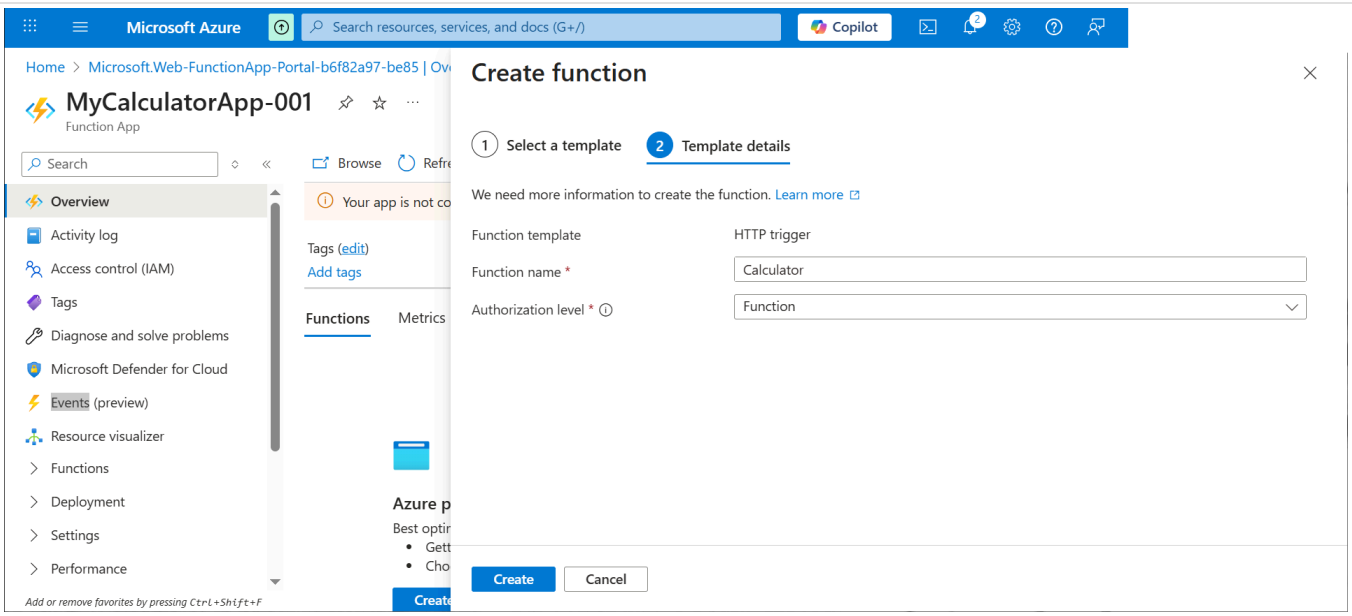


Місце для скріншота: Створення Function App

5. Коли розгортання завершиться, натисніть **"Go to resource"**.
6. У знизу екрана оберіть **"Functions"** -> **"Create in Azure portal"**.
7. Оберіть шаблон **"HTTP trigger"**.
8. **New Function name:** Введіть `Calculator`.
9. **Authorization level:** Оберіть `Function`. (Це означає, що для виклику потрібен буде

секретний ключ).

10. Натисніть **"Create"**.



Місце для скріншота: Створення HTTP Trigger функції

11. Коли функція створиться, натисніть на неї (`Calculator`), а потім у меню зліва оберіть **"Code + Test"**.
12. Видаліть весь код у файлі `index.js` і вставте замість нього наступний код. Цей код (на відміну від AWS) очікує дані лише у `req.body` (тобто `POST` запит).

```
module.exports = async function (context, req) {
    context.log('Calculator function processing a POST request.');

    // Ми очікуємо дані лише у тілі запиту (req.body)
    if (!req.body || req.body.a === undefined || req.body.b === undefined ||
req.body.op === undefined) {
        context.res = {
            status: 400,
            body: "400 Invalid Input: Please provide a JSON body with 'a',
'b', and 'op'."
        };
        return;
    }

    const { a, b, op } = req.body;

    let res = {};
    res.a = Number(a);
    res.b = Number(b);
```

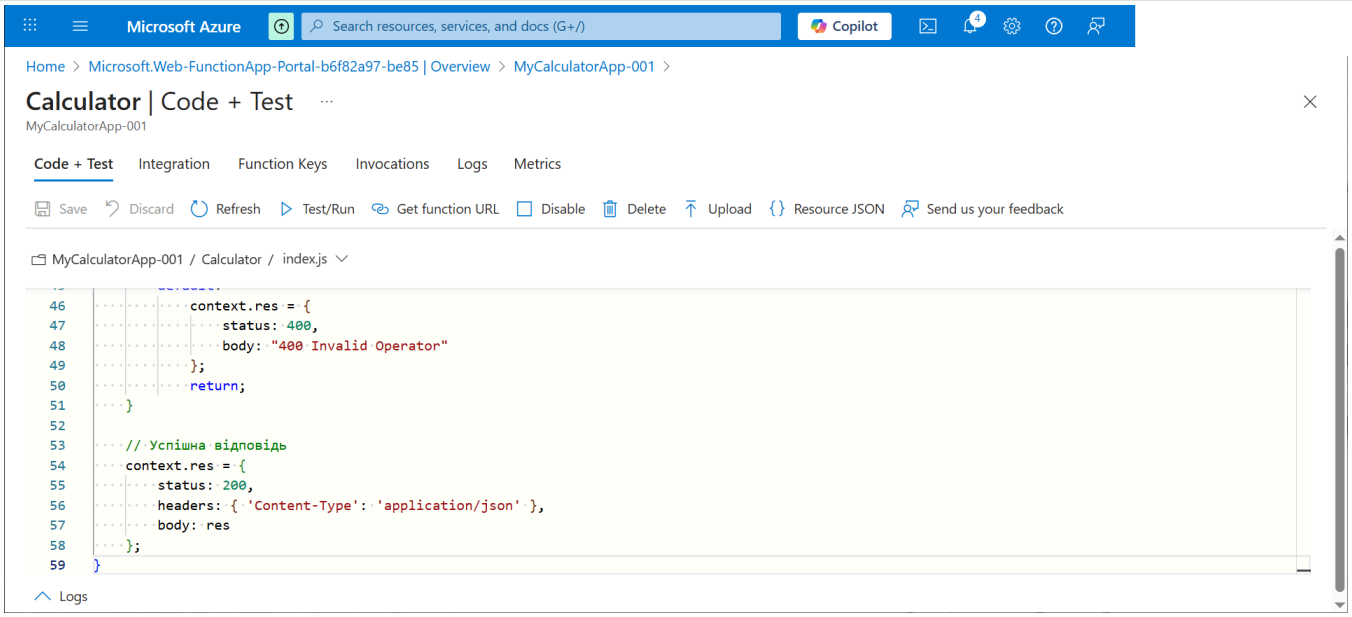
```
    res.op = op;

    if (isNaN(res.a) || isNaN(res.b)) {
        context.res = {
            status: 400,
            body: "400 Invalid Operand"
        };
        return;
    }

    switch(res.op) {
        case "+":
        case "add":
            res.c = res.a + res.b;
            break;
        case "-":
        case "sub":
            res.c = res.a - res.b;
            break;
        case "*":
        case "mul":
            res.c = res.a * res.b;
            break;
        case "/":
        case "div":
            res.c = (res.b === 0) ? NaN : res.a / res.b;
            break;
        default:
            context.res = {
                status: 400,
                body: "400 Invalid Operator"
            };
            return;
    }

    // Успішна відповідь
    context.res = {
        status: 200,
        headers: { 'Content-Type': 'application/json' },
        body: res
    };
}
```

13. Натисніть **"Save"**.



Місце для скріншота: Вкладка "Code + Test" зі вставленим кодом

Крок 2: Тестування функції-калькулятора

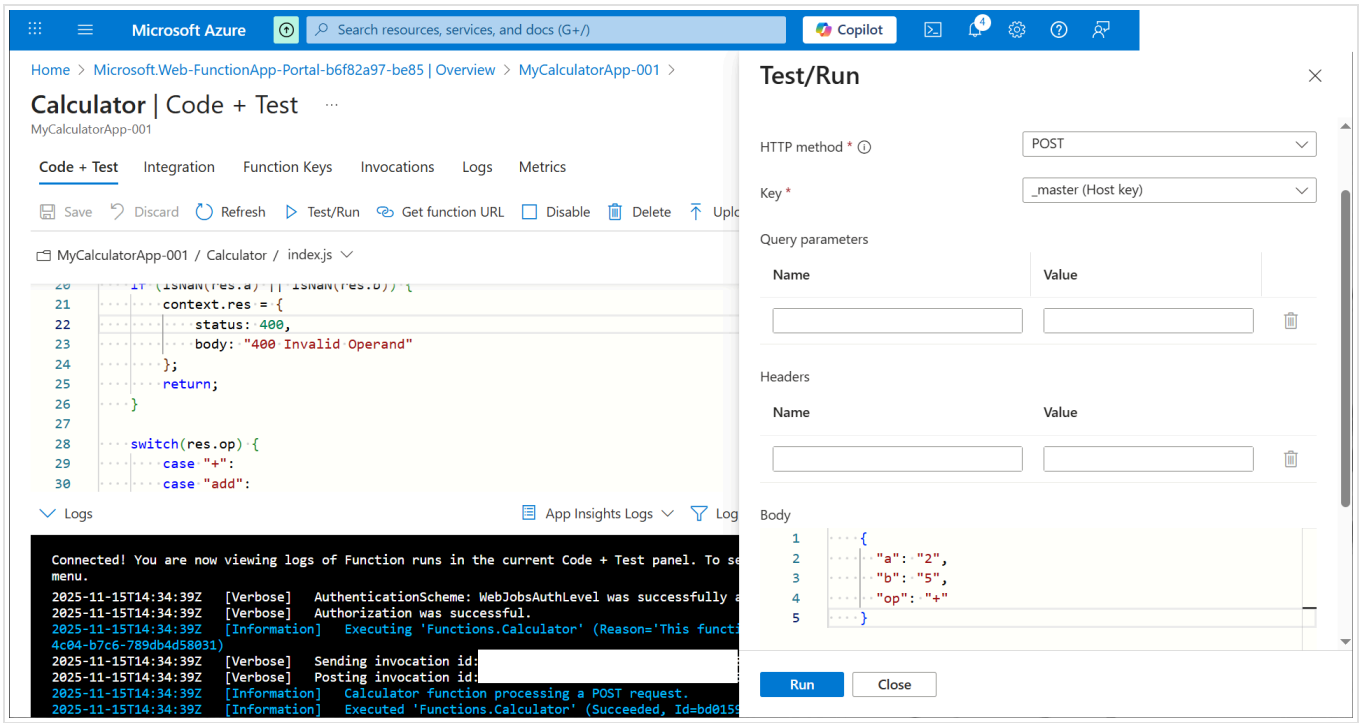
Перевіримо, чи працює наша функція, надіславши їй прямий `POST` запит.

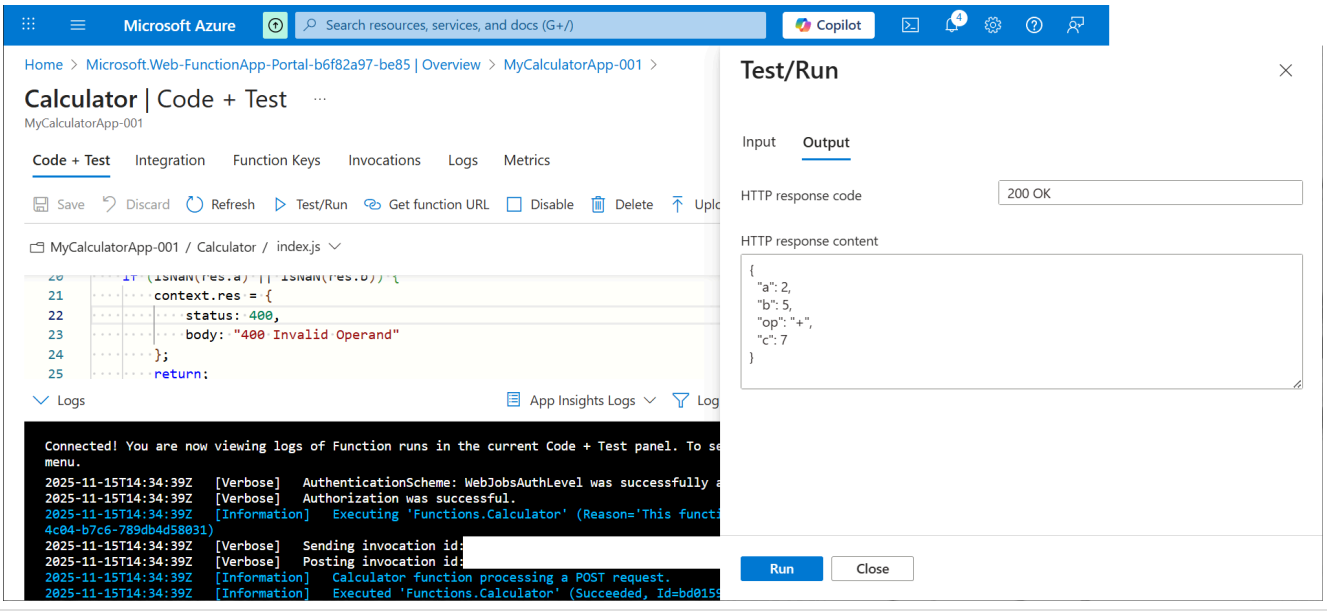
1. На тій самій вкладці **"Code + Test"** натисніть **"Test/Run"**.
2. У правій панелі, що відкрилася:
   - **HTTP method:** `POST`.
   - Перейдіть до **"Body"**.
   - Вставте той самий JSON, що й у лабораторній з AWS:

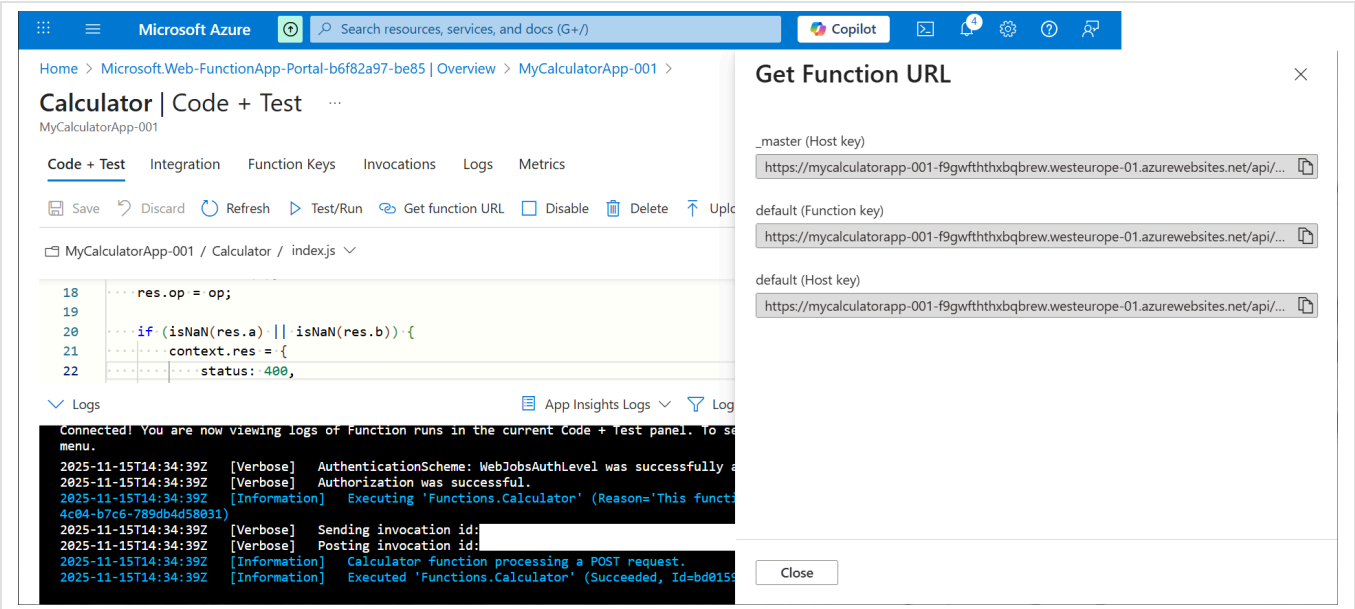
```
{
  "a": "2",
  "b": "5",
  "op": "+"
}
```

3. Натисніть **"Run"**.
4. У вікні **"Output"**, ви маєте побачити результат `HTTP response content` зі статусом `200 OK` та тілом: `{ "a": 2, "b": 5, "op": "+", "c": 7 }` (тіло треба буде потягнути за куточок знизу-справа, щоб побачити повністю).
5. **Збережіть URL функції:**
   - Натисніть кнопку **"Get Function Url"** (отримати URL функції).

- Скопіюйте URL `default (Function key)` (він міститиме `...&code=...`). Збережіть його у Блокноті. Він знадобиться нам для Кроку 4.







Місце для скріншота: Успішний результат тестування функції

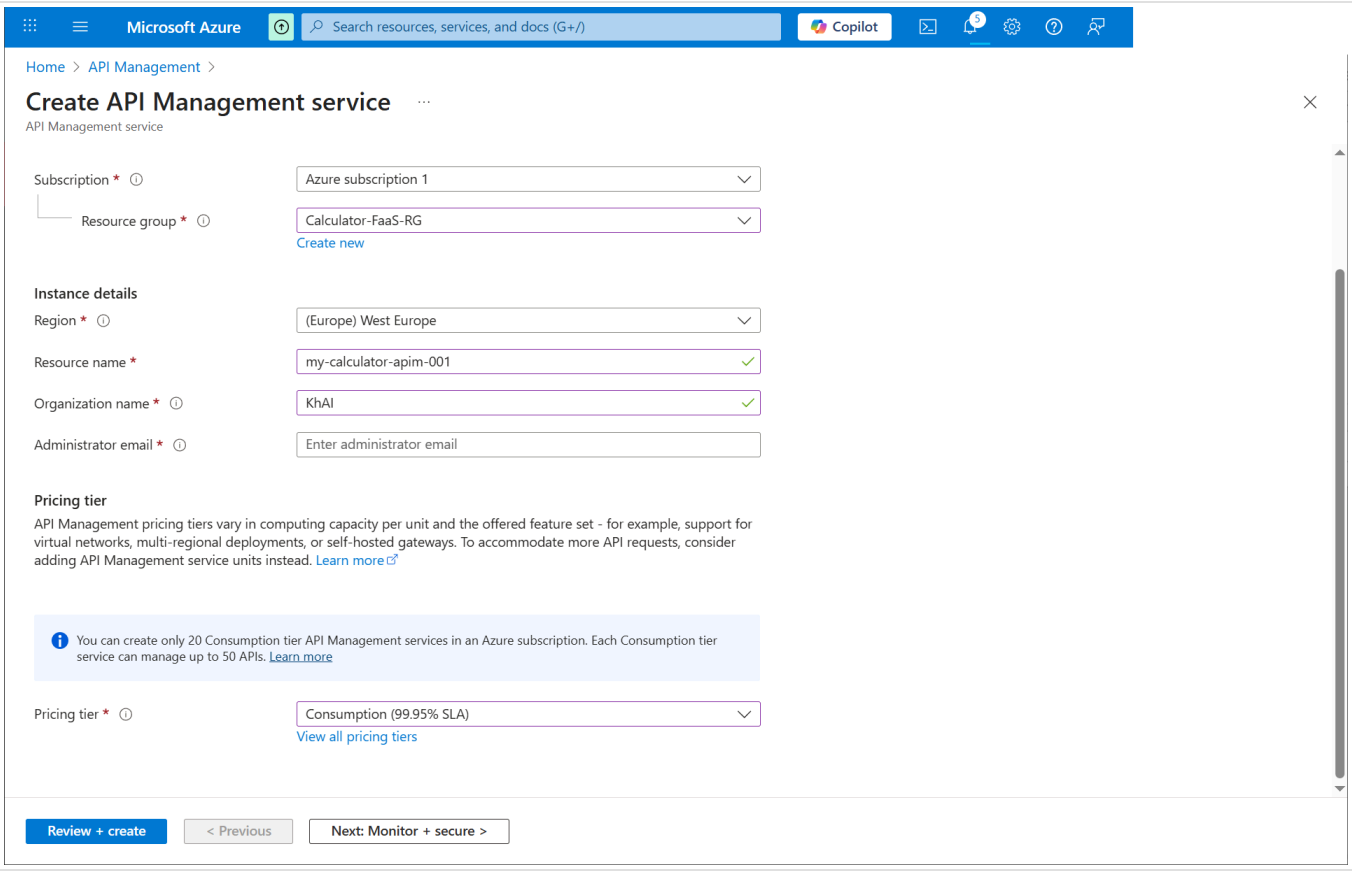
Крок 3: Створення Azure API Management (Шлюзу API)

Тепер ми створимо сервіс (аналог AWS API Gateway), який буде нашою публічною точкою входу.

1. На порталі Azure натисніть **"Create a resource"** (Створити ресурс) -> **"API Management"**.
2. На вкладці **"Basics"** заповніть поля:
   - **Resource Group:** Оберіть вашу групу `Calculator-FaaS-RG`.
   - **Region:** Оберіть той самий регіон, де знаходиться ваша Function App.
   - **Resource name:** Введіть унікальне ім'я, наприклад, `my-calculator-apim` (додайте унікальні цифри).
   - **Organization name:** Назва вашої організації.
   - **Pricing tier:** Оберіть `Consumption (99.95% SLA)`. Це безсерверний, швидкий у розгортанні та дешевий варіант (аналог FaaS).

     > **ВАЖЛИВО:** *Не* обирайте `Developer` або `Basic`, оскільки їх розгортання триває **30-60 хвилин**. Рівень `Consumption` буде готовий за 2-3 хвилини.

3. Натисніть **"Review + create"**, а потім **"Create"**.

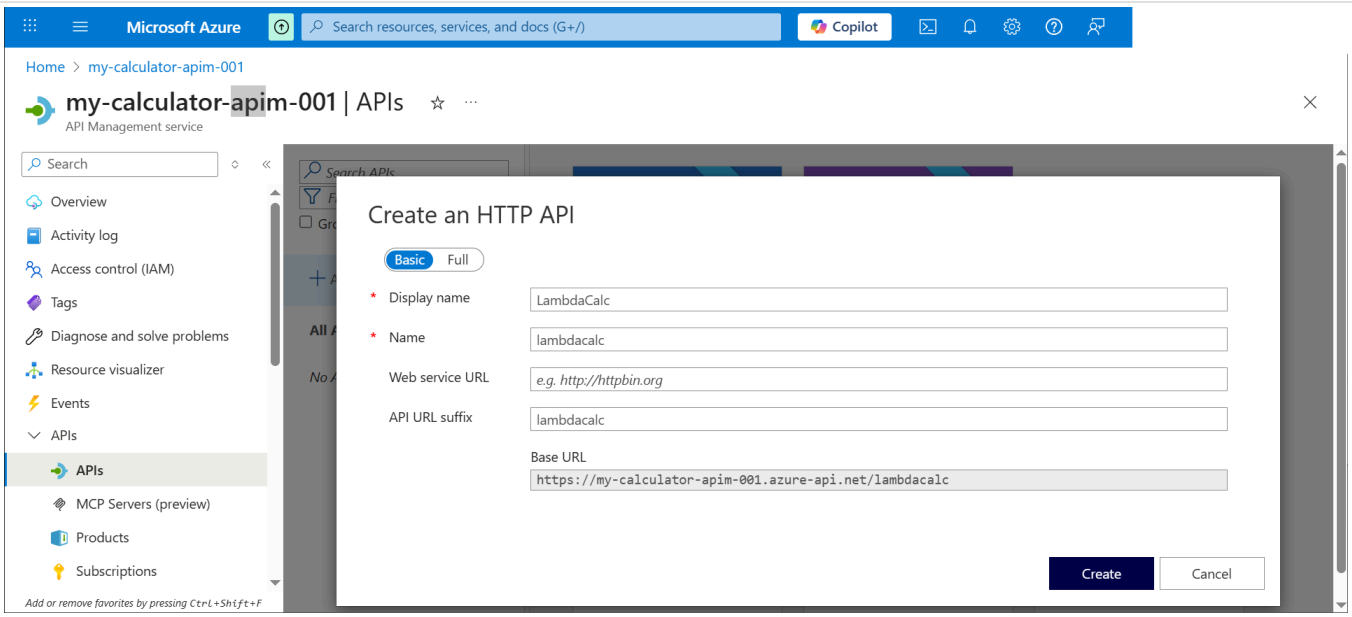


Місце для скріншота: Створення Azure API Management (APIM)

Крок 4: Створення HTTP-методу та перетворення запиту

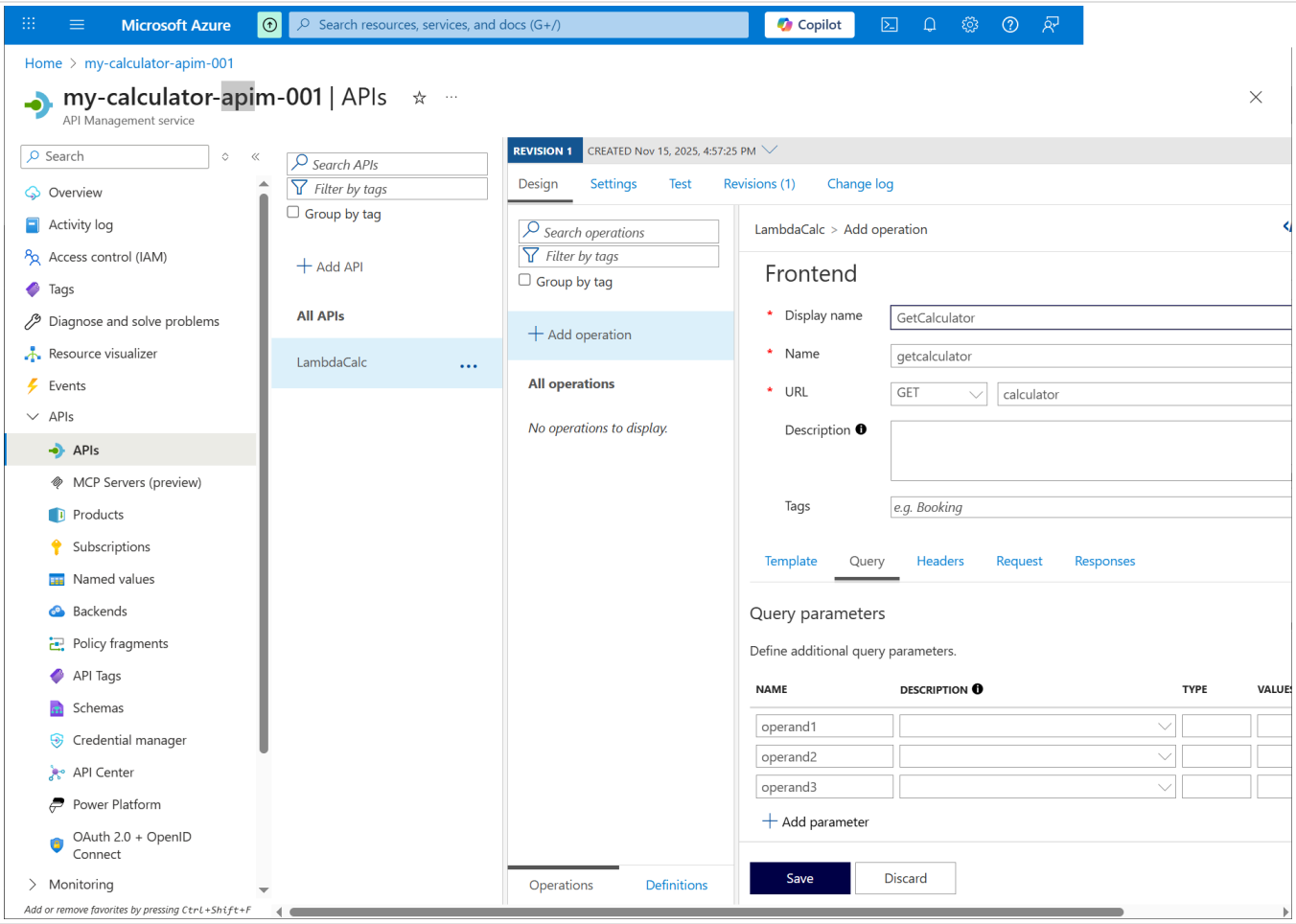
Це аналог Кроку 5 з AWS. Ми створимо `GET` метод, який буде приймати параметри з URL, але перетворювати їх на `POST` запит з JSON-тілом для нашої Azure Function.

1. Коли APIM розгорнеться, перейдіть до нього.
2. У меню зліва оберіть **"APIs"** -> **"APIs"** -> **"+ Add API"** -> **"HTTP"**.
3. У вікні "Create an HTTP API" введіть:
   - **Display name:** `LambdaCalc` (як в AWS lab)
   - **Name:** (Залиште `lambdacalc`)
   - **API URL suffix:** `lambdacalc`
4. Натисніть **"Create"**.



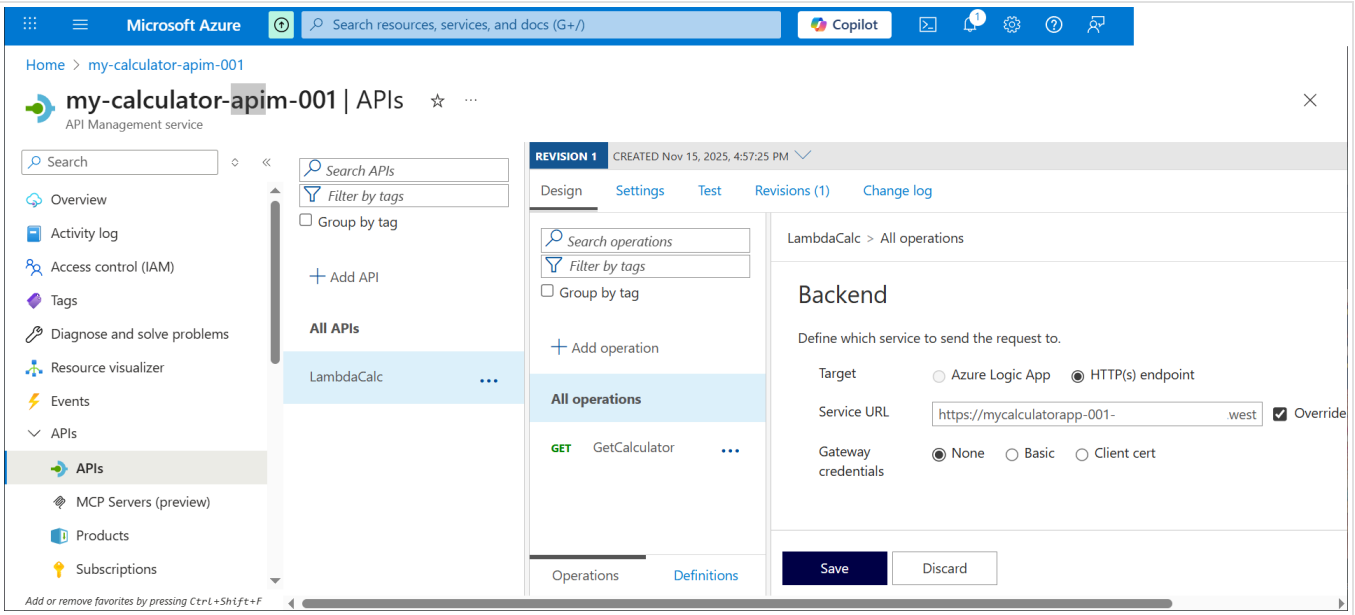
*Місце для скріншота: Створення HTTP API

5. Тепер натисніть на **"+ Add Operation"** (Додати операцію).
6. На вкладці **"Frontend"** (це те, що бачить клієнт) налаштуйте `GET` запит:
   - **Display name:** `GetCalculator`
   - **URL:** `GET` / `calculator`
   - Перейдіть на під-вкладку **"Query"** (внизу).
   - Натисніть **"+ Add query parameter"** 3 рази, щоб додати:
     1. `operand1`
     2. `operand2`
     3. `operator`
   - Натисніть **"Save"**.



Місце для скріншота: Налаштування "Frontend" з параметрами запиту (Query)

7. Тепер у секції **"Backend"** (це те, куди йде запит) вкажіть вашу Azure Function:
   - **Target:** `HTTP(s) endpoint`
   - **Service URL:** Натисніть галочку поряд з `Override` та вставте сюди **URL вашої функції, ОБРІЗАВШИ ключ ?code=...**, який ви скопіювали у **Кроці 2 (пункт 5)**.
     Наприклад: `https://mycalculatorapp-123.azurewebsites.net/api/Calculator`



Місце для скріншота: Налаштування "Backend"

8. Тепер найголовніше — **Перетворення Запиту**. Перейдіть до секції **"Inbound processing"** (Обробка вхідних запитів) і натисніть на іконку `</>`.
9. Відкриється редактор XML-політик. Замініть вміст тегу `<inbound>...</inbound>` на цей код, **ЗАМІНИВШИ [ВАШ_ДОВГИЙ_КЛЮЧ_ФУНКЦІЇ_З_КРОКУ_2.5]** на ключ, що йде після ?code=, у посиланні, що ви скопіювали у кроці 2.5.

```
<policies>
    <inbound>
        <base />

        <set-method>POST</set-method>

        <set-header name="Content-Type" exists-action="override">
            <value>application/json</value>
        </set-header>

        <set-body>@{
            return new JObject(
                new JProperty("a",
context.Request.Url.Query.GetValueOrDefault("operand1", "0")),
                new JProperty("b",
context.Request.Url.Query.GetValueOrDefault("operand2", "0")),
                new JProperty("op",
context.Request.Url.Query.GetValueOrDefault("operator", "+"))
            ).ToString();
        }</set-body>
```

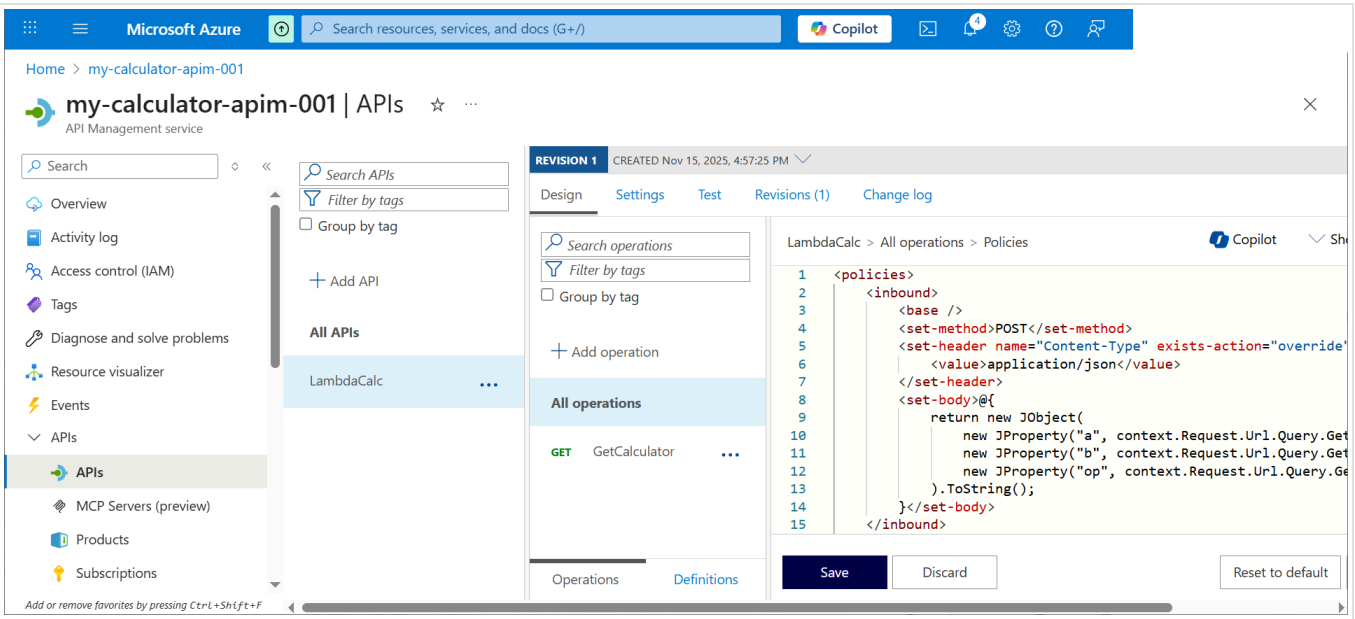
```
        <set-backend-service base-
url="https://[ВАШЕ_ІМ'Я_APP].azurewebsites.net" />

        <rewrite-uri template="/api/Calculator" />

        <set-query-parameter name="code" exists-action="override">
            <value>[ВАШ_ДОВГИЙ_КЛЮЧ_ФУНКЦІЇ_З_КРОКУ_2.5]</value>
        </set-query-parameter>

    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

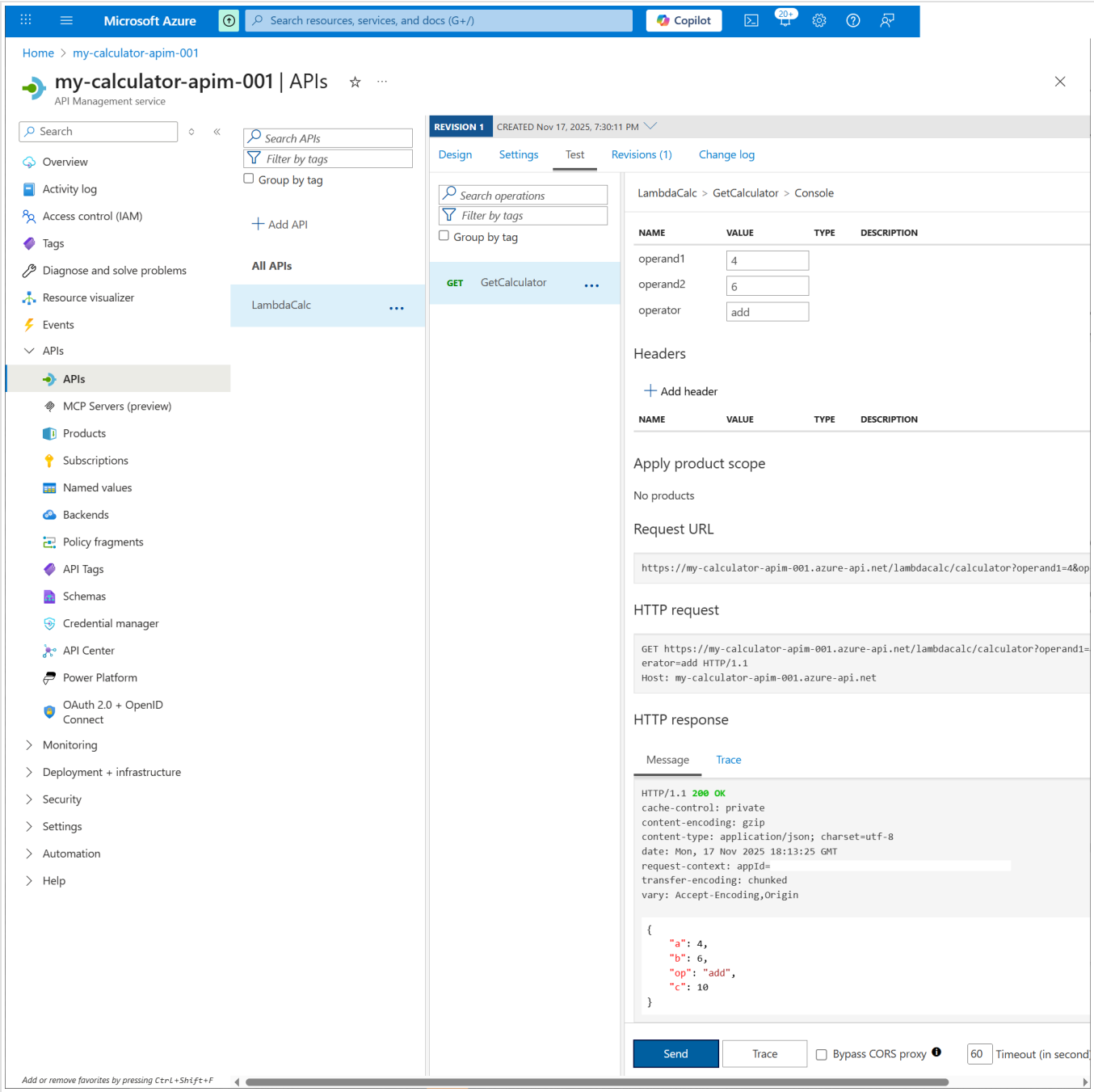
10. Натисніть **"Save"**.



Місце для скріншота: Редактор політик "Inbound processing" з XML-кодом

Крок 5: Розгортання та тестування API

1. У вашому APIM перейдіть на вкладку **"Test"** (Тест).
2. Оберіть операцію `GET /calculator`.
3. Прокрутіть до секції **"Query parameters"** (Параметри запиту).
4. Введіть тестові дані:
   - `operand1` : `4`
   - `operand2` : `6`
   - `operator` : `add`
5. Натисніть **"Send"**.
6. Якщо все налаштовано правильно, ви маєте побачити внизу `HTTP response` **200 OK** та JSON-тіло:
   `{ "a": 4, "b": 6, "op": "add", "c": 10 }`

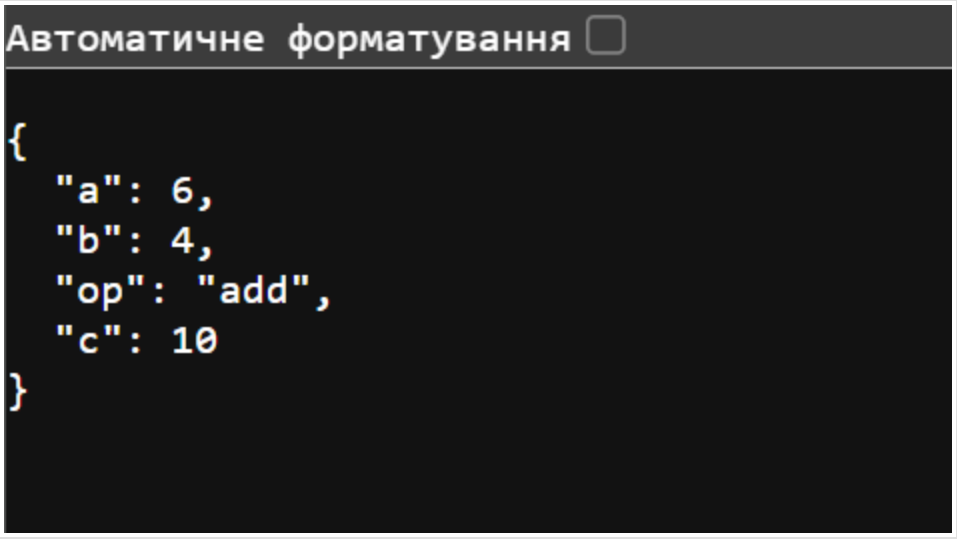


Місце для скріншота: Тестування APIM на вкладці "Test"

7. Тепер протестуємо API "ззовні" (як у Кроці 6 AWS).
8. Перейдіть на вкладку **"Settings"** (Налаштування) вашого API (`LambdaCalc`) та скопіюйте **"Gateway URL"**. Він матиме вигляд `https://my-calculator-apim.azure-api.net/lambdacalc` (Також, можна скопіювати одразу з параметрами, після тестування у полі `Request URL`).
9. У вкладці **"Settings"**, приберіть галочку навпроти `Subscription` -> `Subscription required` (інакше, видаватиме помилку 401).
10. Скомбінуйте цей URL з вашим ресурсом (`/calculator`) та параметрами. Вставте отриманий рядок у нову вкладку браузера (або у SoapUI):

`https://my-calculator-apim.azure-api.net/lambdacalc/calculator?operand1=4&operand2=6&operator=add`

11. Ви маєте побачити той самий JSON-результат у браузері.
    `{ "a": 4, "b": 6, "op": "add", "c": 10 }`

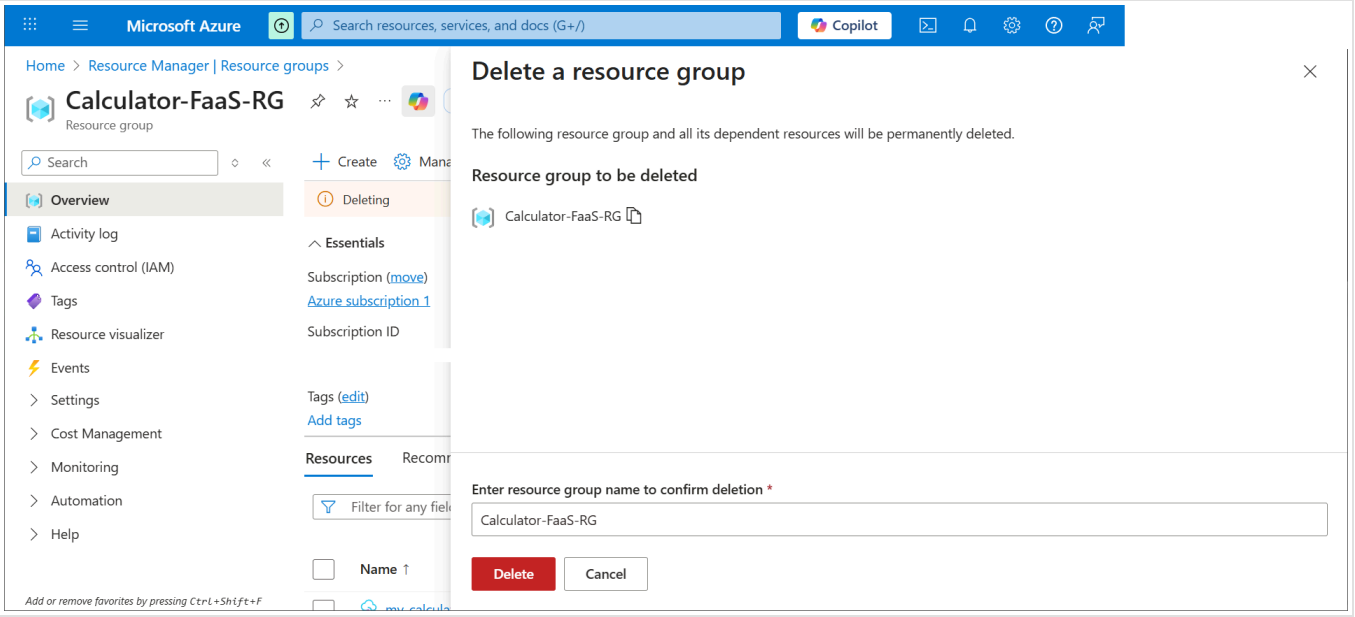


Місце для скріншота: Тестування APIM у вікні браузера

Крок 6: Видалення ресурсів

Щоб уникнути будь-яких витрат, видаліть усі створені ресурси. Найпростіший спосіб в Azure — видалити групу ресурсів, яка містить усе.

1. На головній сторінці порталу Azure перейдіть до **"Resource groups"**.
2. Знайдіть вашу групу `Calculator-FaaS-RG`.
3. Натисніть на неї, а потім натисніть **"Delete resource group"**.
4. Введіть назву групи для підтвердження та натисніть **"Delete"**. Це видалить і Function App, і API Management.



Місце для скріншота: Видалення Resource Group