

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.

Create Function App (Consumption)

...

[Basics](#) [Networking](#) [Monitoring](#) [Deployment](#) [Authentication](#) [Tags](#) [Review + create](#)

Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Azure subscription 1

Resource Group * ⓘ

(New) Calculator-FaaS-RG

[Create new](#)

Instance Details

Function App name *

MyCalculatorApp-001

-f9gwftthxbqbrew.westeurope-01.azurewebsites.net

Secure unique default hostname on. [More about this update ↗](#)

Operating System *



Linux (legacy)



Windows

Flex Consumption is now the recommended serverless hosting plan for Azure Functions. [Learn more.](#)

Runtime stack *

Node.js

Version *

22 LTS

Region *

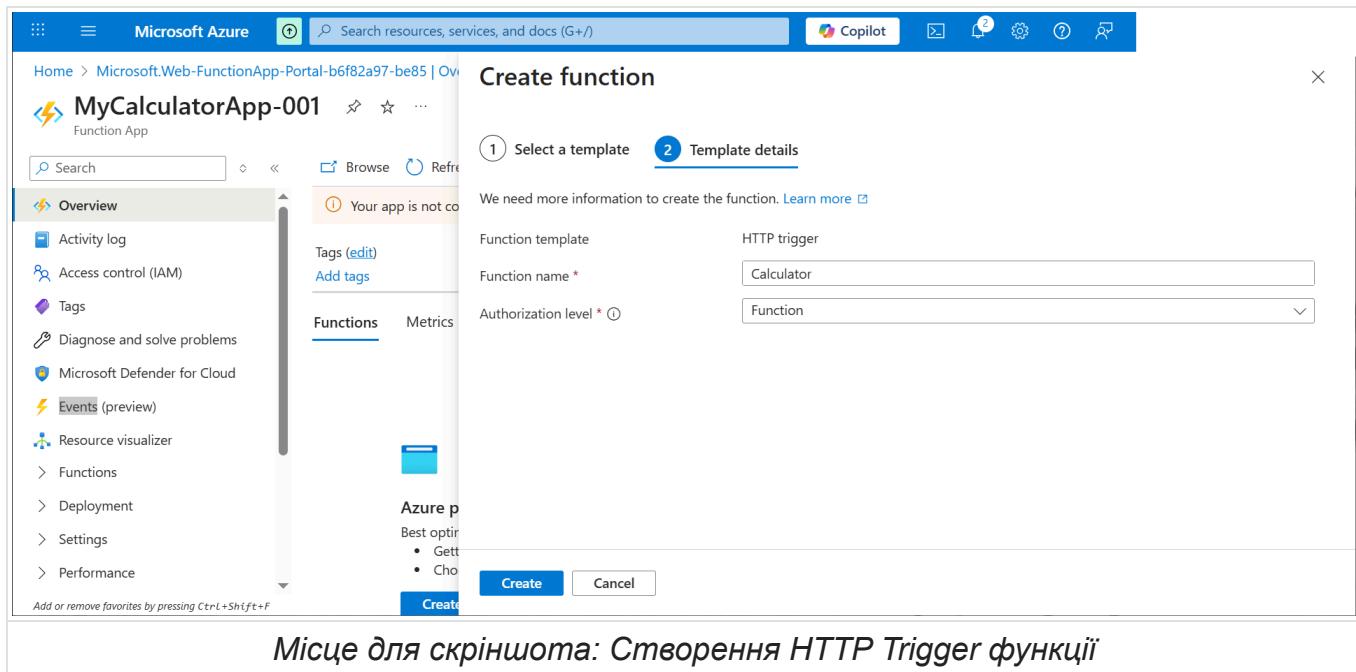
West Europe

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".



MyCalculatorApp-001

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Events (preview)

Resource visualizer

> Functions

> Deployment

> Settings

> Performance

Add or remove favorites by pressing **Ctrl+Shift+F**

Create function

1 Select a template 2 Template details

We need more information to create the function. [Learn more](#)

Function template: HTTP trigger

Function name: Calculator

Authorization level: Function

Azure p
Best optim:
• Gett
• Cho

Create Cancel

Mісце 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".

The screenshot shows the Microsoft Azure Function App portal. In the top navigation bar, it says "Microsoft Azure" and "Search resources, services, and docs (G+/-)". Below the navigation bar, the URL is "Home > Microsoft.Web.FunctionApp-Portal-b6f82a97-be85 | Overview > MyCalculatorApp-001 >". The main title is "Calculator | Code + Test". Underneath, it says "MyCalculatorApp-001". There are tabs for "Code + Test" (which is selected), "Integration", "Function Keys", "Invocations", "Logs", and "Metrics". Below the tabs are buttons for "Save", "Discard", "Refresh", "Test/Run", "Get function URL", "Disable", "Delete", "Upload", "Resource JSON", and "Send us your feedback". The main content area shows the file "index.js" with the following code:

```
46 context.res = {
47   status: 400,
48   body: "400 Invalid Operator"
49 };
50 return;
51 }
52
53 // Успішна відповідь
54 context.res = {
55   status: 200,
56   headers: { 'Content-Type': 'application/json' },
57   body: res
58 };
59 }
```

At the bottom left, there's a "Logs" link. At the bottom right, a note says "Місце для скріншота: Вкладка "Code + Test" зі вставленим кодом".

Крок 2: Тестування функції-калькулятора

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

- На тій самій вкладці "**Code + Test**" натисніть "**Test/Run**".
- 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": "+"
}
```

- Натисніть "**Run**".
- 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 куточек знизу-справа, щоб побачити повністю).
- 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.

The screenshot shows the Microsoft Azure Function App Test/Run interface. The URL is `https://MyCalculatorApp-001.azurewebsites.net/api/Calculator?code=4c04-b7c6-789db4d58831`. The request method is set to POST, and the key is master (Host key). The body contains the JSON `{"a": "2", "b": "5", "op": "+"}`. The response status is 200 OK, and the response content is `{"c": 7}`.

```

HTTP method * POST
Key * master (Host key)

Name Value
[empty]
[empty]

Name Value
[empty]
[empty]

Body
1 ...
2 ...
3 ...
4 ...
5 ...

Run Close

```

The screenshot shows the Microsoft Azure Function App Test/Run interface. The URL is `https://MyCalculatorApp-001.azurewebsites.net/api/Calculator?code=4c04-b7c6-789db4d58831`. The request method is set to POST, and the key is master (Host key). The body contains the JSON `{"a": "2", "b": "5", "op": "+"}`. The response status is 200 OK, and the response content is `{"c": 7}`.

```

Input Output
HTTP response code 200 OK
HTTP response content
{
  "a": 2,
  "b": 5,
  "op": "+",
  "c": 7
}

Run Close

```

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.

- На порталі Azure натисніть "**Create a resource**" (Створити ресурс) -> "**API Management**".
- 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.

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

Subscription * ① Azure subscription 1

Resource group * ① Calculator-FaaS-RG Create new

Instance details

Region * ① (Europe) West Europe

Resource name * my-calculator-apim-001

Organization name * ① KhAI

Administrator email * ① Enter administrator email

Pricing tier

API Management pricing tiers vary in computing capacity per unit and the offered feature set - for example, support for virtual networks, multi-regional deployments, or self-hosted gateways. To accommodate more API requests, consider adding API Management service units instead. [Learn more](#)

💡 You can create only 20 Consumption tier API Management services in an Azure subscription. Each Consumption tier service can manage up to 50 APIs. [Learn more](#)

Pricing tier * ① Consumption (99.95% SLA) View all pricing tiers

Review + create < Previous Next: Monitor + secure >

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".

The screenshot shows the Microsoft Azure API Management service interface. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Events, and APIs. The APIs section is currently selected. A modal window titled "Create an HTTP API" is open in the center. It has two tabs: "Basic" (which is selected) and "Full". Under "Basic", there are fields for "Display name" (set to "LambdaCalc"), "Name" (set to "lambdacalc"), "Web service URL" (set to "e.g. http://httpbin.org"), "API URL suffix" (set to "lambdacalc"), and "Base URL" (set to "https://my-calculator-apim-001.azure-api.net/lambdacalc"). At the bottom of the modal are "Create" and "Cancel" buttons. The main page background shows a search bar at the top and a list of APIs on the left.

*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**".

The screenshot shows the Microsoft Azure API Management service interface. On the left, the navigation bar includes 'Home > my-calculator-apim-001' and a search bar. The main area displays the 'my-calculator-apim-001 | APIs' page for an 'API Management service'. A sidebar on the left lists various management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Events, APIs, and more. The 'APIs' section is currently selected. The main content area shows a 'REVISION 1' entry created on Nov 15, 2025, at 4:57:25 PM. The 'Design' tab is selected, showing the 'Frontend' configuration for the 'LambdaCalc > Add operation'. The configuration includes:

- Display name: GetCalculator
- Name: getcalculator
- URL: GET /calculator
- Description: (empty)
- Tags: e.g. Booking

Below the configuration are tabs for 'Template', 'Query' (which is selected), 'Headers', 'Request', and 'Responses'. The 'Query parameters' section defines three parameters: 'operand1', 'operand2', and 'operand3', each with a dropdown menu. At the bottom are 'Save' and 'Discard' buttons.

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>

The screenshot shows the Microsoft Azure API Management service interface. On the left, there's a navigation sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Events, APIs, MCP Servers (preview), Products, and Subscriptions. The 'APIs' section is selected. In the main area, a card for 'my-calculator-apim-001 | APIs' is shown. Below it, a card for 'LambdaCalc' is selected. The right side shows the 'Backend' configuration for the 'GetCalculator' operation. It includes fields for Target (set to 'HTTP(s) endpoint'), Service URL ('https://mycalculatorapp-001-west'), and Gateway credentials ('None'). Buttons for 'Save' and 'Discard' are at the bottom.

Місце для скріншота: Налаштування "Backend"

8. Тепер найголовніше — **Перетворення Запиту**. Переїдіть 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>
    </inbound>
</policies>

```

```

<set-backend-service base-
url="https://[ВАШ_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".

The screenshot shows the Microsoft Azure API Management service interface. The left sidebar has a 'my-calculator-apim-001 | APIs' section selected. Under 'APIs', 'LambdaCalc' is highlighted. The main area shows a policy editor for the 'LambdaCalc' API. The 'Design' tab is selected, displaying the XML code for inbound processing:

```

<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.Get("a")),
                new JProperty("b", context.Request.Url.Query.Get("b")),
                new JProperty("op", context.Request.Url.Query.Get("code"))
            ).ToString();
        }</set-body>
    </inbound>

```

Below the XML code are 'Save', 'Discard', and 'Reset to default' buttons.

Mісце 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 }
```

Microsoft Azure | Search resources, services, and docs (G+)

Home > my-calculator-apim-001

my-calculator-apim-001 | APIs

API Management service

Search APIs

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Resource visualizer

Events

APIs

APIs (selected)

- MCP Servers (preview)
- Products
- Subscriptions
- Named values
- Backends
- Policy fragments
- API Tags
- Schemas
- Credential manager
- API Center
- Power Platform
- OAuth 2.0 + OpenID Connect

REVISION 1 CREATED Nov 17, 2025, 7:30:11 PM

Design Settings Test Revisions (1) Change log

Search operations

Filter by tags

Group by tag

GET GetCalculator ...

LambdaCalc > GetCalculator > Console

NAME	VALUE	TYPE	DESCRIPTION
operand1	4		
operand2	6		
operator	add		

Headers

Add header

NAME	VALUE	TYPE	DESCRIPTION
------	-------	------	-------------

Apply product scope

No products

Request URL

https://my-calculator-apim-001.azure-api.net/lambdacalc/calculator?operand1=4&operator=add

HTTP request

```
GET https://my-calculator-apim-001.azure-api.net/lambdacalc/calculator?operand1=4&operator=add HTTP/1.1
Host: my-calculator-apim-001.azure-api.net
```

HTTP response

Message Trace

HTTP/1.1 200 OK

cache-control: private

content-encoding: gzip

content-type: application/json; charset=utf-8

date: Mon, 17 Nov 2025 18:13:25 GMT

request-context: appId=

transfer-encoding: chunked

vary: Accept-Encoding,Origin

```
{
  "a": 4,
  "b": 6,
  "op": "add",
  "c": 10
}
```

Send Trace Bypass CORS proxy Timeout (in seconds)

Add or remove favorites by pressing Ctrl+Shift+F

Mісце 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](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 }
```

Автоматичне форматування

```
{  
    "a": 6,  
    "b": 4,  
    "op": "add",  
    "c": 10  
}
```

Mісце 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.

Microsoft Azure Search resources, services, and docs (G+/)

Home > Resource Manager | Resource groups >

Calculator-FaaS-RG Resource group

Delete a resource group

The following resource group and all its dependent resources will be permanently deleted.

Resource group to be deleted

Calculator-FaaS-RG

Subscription (move) [Azure subscription 1](#)

Subscription ID

Tags (edit) [Add tags](#)

Resources Recom

Enter resource group name to confirm deletion *

Calculator-FaaS-RG

Name ↑ my calculate

Delete **Cancel**

Add or remove favorites by pressing **Ctrl+Shift+F**

Mісце для скріншота: Видалення Resource Group

