Лекція 5. Введення у сховища даних, *Azure Queue Storage*

Тема 1. Основні компоненти Azure Queue Storage: Storage Account, Queue та Message.

Тема 2. Рішення для створення автоматизованого робочого процесу отримання повідомлень з Azure Queue Storage — Azure Logic Apps.

Тема 3. Python SDK для роботи з Azure Queue Storage.

Azure Queue Storage

Azure Queue Storage — це хмарний сервіс від Microsoft, призначений для зберігання великих обсягів повідомлень, дозволяє асинхронно обробляти та управляти великими даними через черги, а також дає змогу зберігати й обробляти повідомлення з будь-якої точки світу.
Azure Queue Storage — це компонент Azure Storage, який забезпечує функціональність черг для зберігання повідомлень.
Розмір повідомлення: Одне повідомлення може мати розмір до 64 KB.
Ємність черги: Черга може містити мільйони повідомлень, що обмежено лише максимальною ємністю сховища, яка визначена для вашого облікового запису.

Застосування: Черги часто використовуються для зберігання завдань для асинхронної обробки. Це допомагає організувати робочі процеси в таких архітектурних стилях, як Web-Queue-Worker.

Основні компоненти

Queue Storage складається з трьох основних елементів: Storage Account, Queue, і Message.
a) Storage Account (Обліковий запис сховища)
Усі операції доступу до Azure Storage здійснюються через обліковий запис сховища. Обліковий запис визначає зберігання даних, а також забезпечує безпеку доступу.
URL для доступу до Queue: Формат URL для доступу до черги наступний:

https://<storage account>.queue.core.windows.net/<queue>

b) Queue (Черга)
Черга містить набір повідомлень. Назва черги повинна бути у нижньому регістрі (нижній регістр для усіх символів).
Правила іменування: Існують правила для іменування черг. Вони мають бути короткими, зрозумілими і містити лише літери і цифри.

c) Message (Повідомлення)
Повідомлення — це дані, які зберігаються в черзі. Вони можуть мати будь-який формат і не перевищувати 64 КВ. Повідомлення можуть мати параметр «Time-to-Live» (TTL), який визначає період їх зберігання в черзі.
TTL (Час життя повідомлення):
До версії 2017-07-29 максимальний TTL складав 7 днів.
Після цієї дати можна вказати будь-яке позитивне значення для TTL або -1, що вказує на те, що повідомлення не має терміну придатності.
Якщо TTL не зазначений, за замовчуванням використовується значення 7 днів.

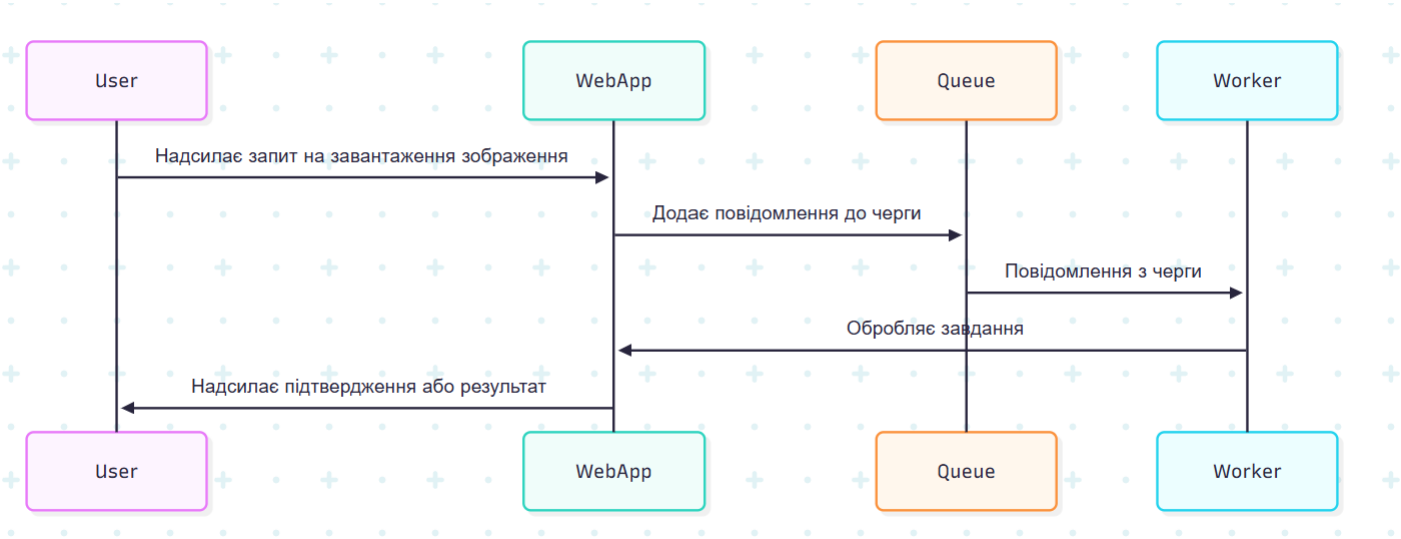
Приклад використання черг

Уявімо, що у вас є веб-додаток для завантаження зображень. Ви хочете створити систему, де кожен запит на завантаження буде додаватися в чергу, і пізніше буде оброблений за допомогою фонової обробки:

1.Користувач надсилає запит на завантаження зображення.

2.Це зображення додається в чергу як повідомлення.

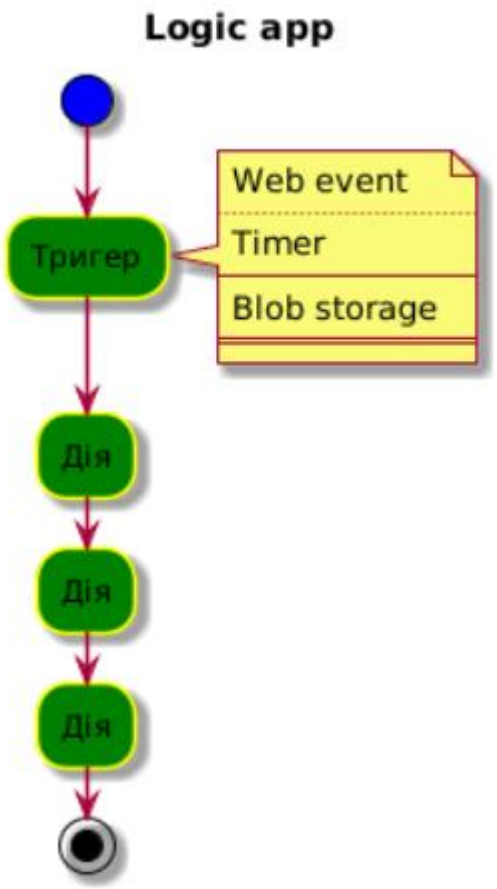
3.Робочий процес, який постійно моніторить цю чергу, отримує завдання і завантажує зображення на сервер або обробляє його іншим чином.



Azure Logic Apps

Рішення, для створення автоматизованого робочого процесу управління даними.

Терміни

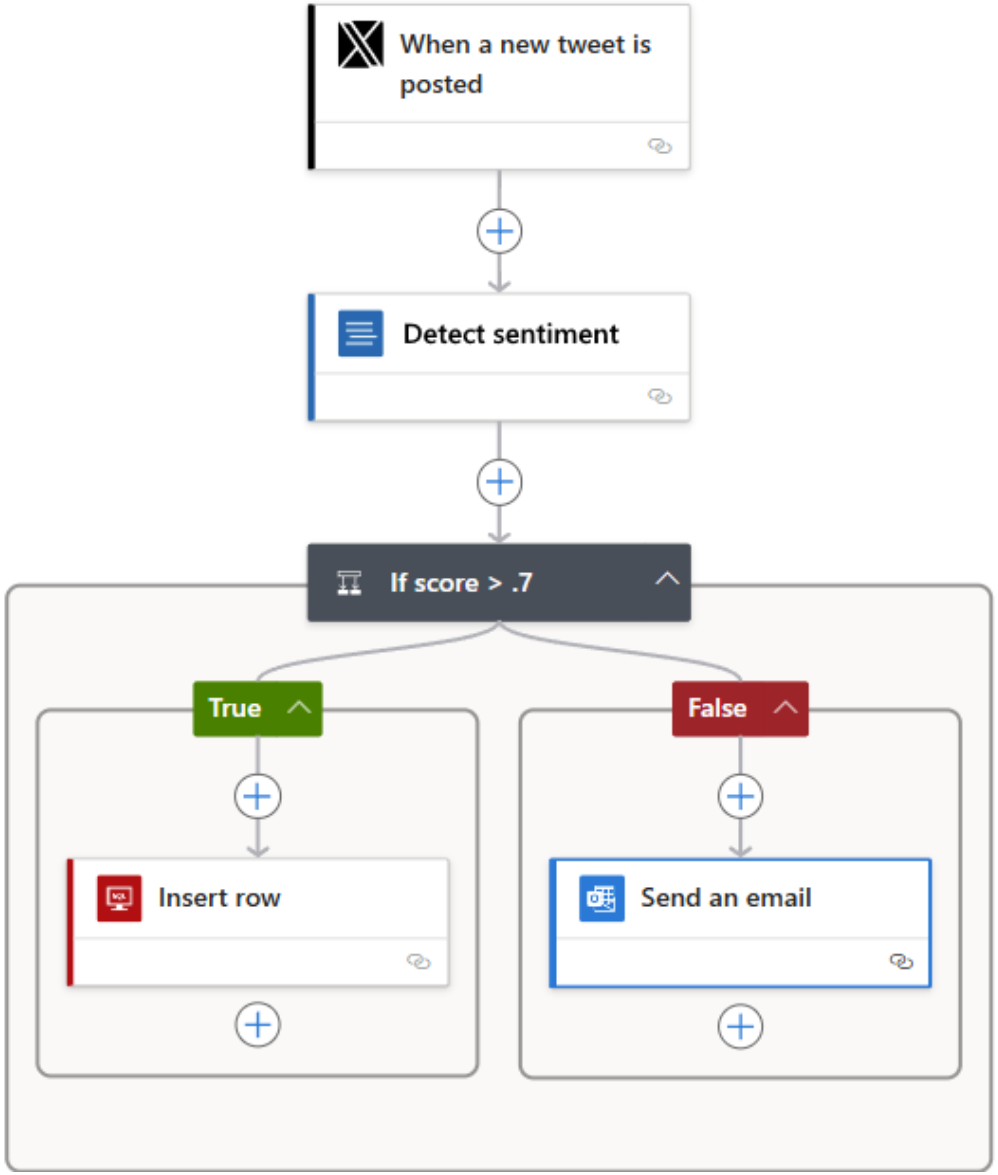


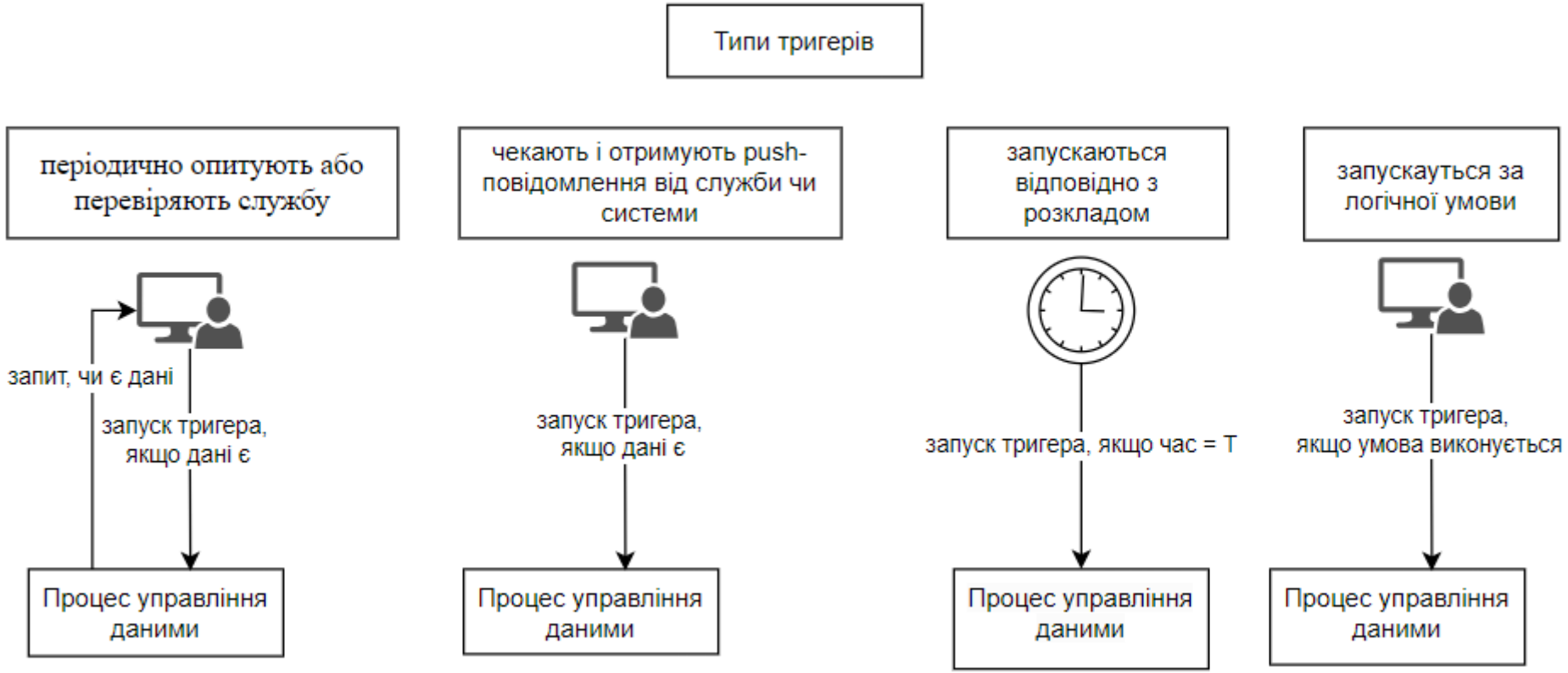
Робочий процес - Ряд операцій, що визначають задачу, бізнес-процес або робочу навантаження. Кожен робочий процес завжди починається з однієї операції тригера, після якої необхідно додати одну або кілька операцій.

Тригер - Перша операція в будь-якому робочому процесі, вказуючи критерії, які необхідно виконати перед виконанням будь-яких наступних операцій у цьому робочому процесі. Наприклад, подія тригера може отримувати повідомлення електронної пошти

З'єднувач — це компонент, який надає інтерфейс до служби або системи у відео операцій. Наприклад, з'єднувач X дозволяє надсилати та завантажувати повідомлення, а з'єднувач Office 365 Outlook дозволяє керувати електронною поштою, календарем і контактами. Програми Azure Logic надають понад 1000 попередньо створених з'єднувачів, які можна використовувати для створення робочих процесів.Соединитель використовує служби REST або SOAP API для виконання фактичної роботи. При використанні з'єднувача в робочому процесі - з'єднувач викликає базову службу API. Таким чином, з'єднувач є в основному оболонкою навколо API.

Конструктор робочих процесів — це графічний інструмент для створення робочих процесів. Конструктор забезпечує область холста, в якій створюється робочий процес шляхом додавання триггера та дій. На наступному знімку екрану показано робочий процес додатків логіки для моніторингу соціальних мереж у конструкторі:

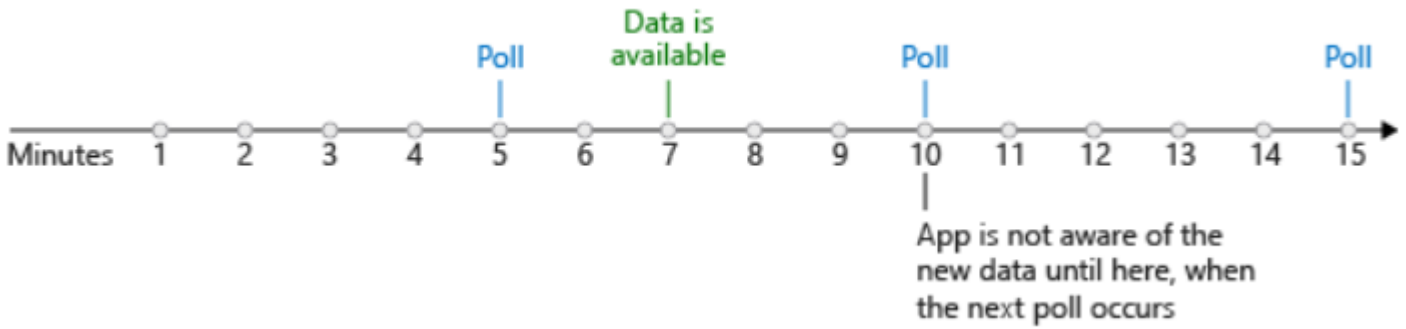




Polling - Тригер, який періодично опитує або перевіряє службу

Push - Тригер, який очікує отримання від служби або системи повідомлень про наявність даних або подій

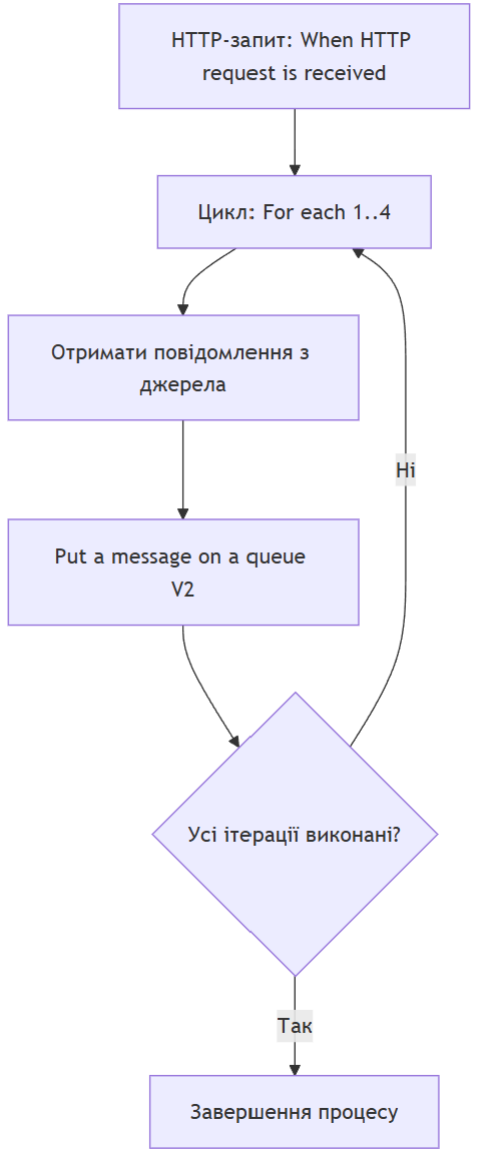
Polling - Тригер, який періодично опитує або перевіряє службу. Для тригера потрібно визначити – частоту та інтервал. У найгіршому випадку можлива затримка виявлення нових даних дорівнює інтервалуитув



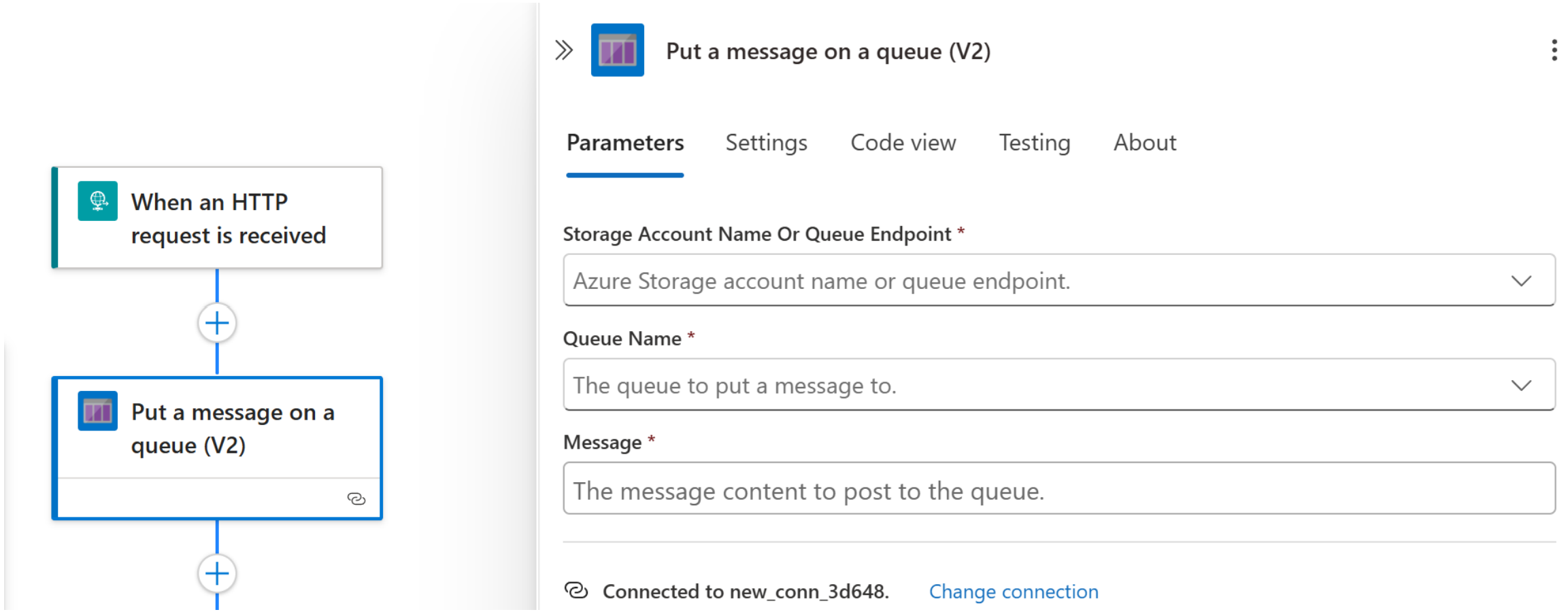
Push - Тригер, який очікує отримання від служби або системи повідомлень про наявність даних або подій, що відповідають певним умовам. Тригер підписується на кінцеву точку у зовнішній службі чи системі. Коли з'являються нові дані або події, що відповідають умовам, служба або система повідомляє тригер, який негайно запускає робочий процес. Наприклад, з'єднувач Службової шини Azure містить тригер, що сповіщає, який спрацьовує при додаванні повідомлення в чергу Службової шини Azure. Push - Тригер не спрацьовують, якщо дані або події відсутні. Тож витрат на опитування немає. З іншого боку, з появою нових даних чи подій такі тригери спрацьовують негайно.

Приклад1. реалізації процесу отримання повідомлення від web доданку і збереження в черзі

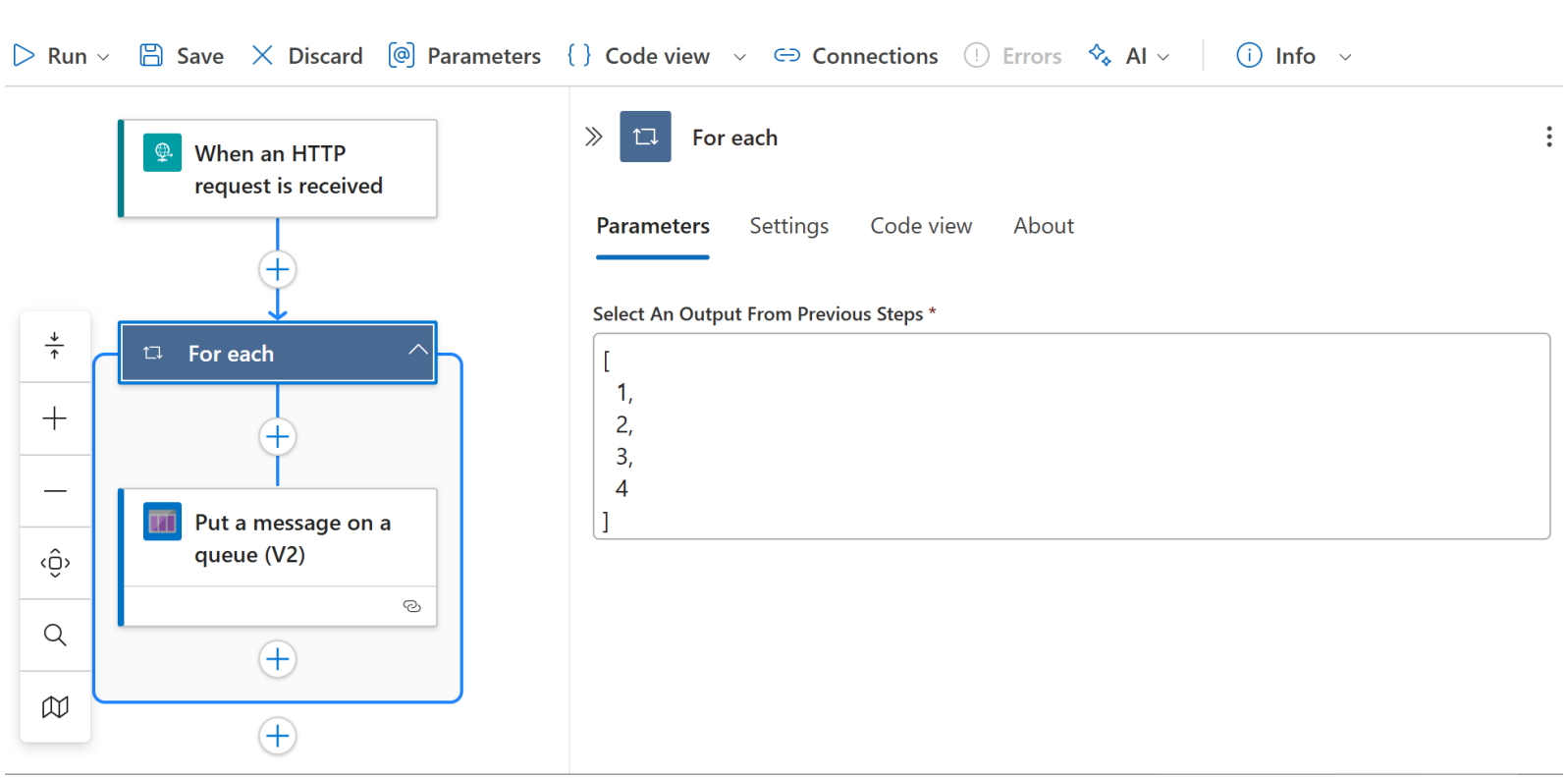
Процесу Logic App, який запускається за допомогою тригера "When HTTP request is received";
у циклі 4 рази виконує дію "Put a message on a queue (V2)", щоб додати повідомлення в Azure Queue Storage.

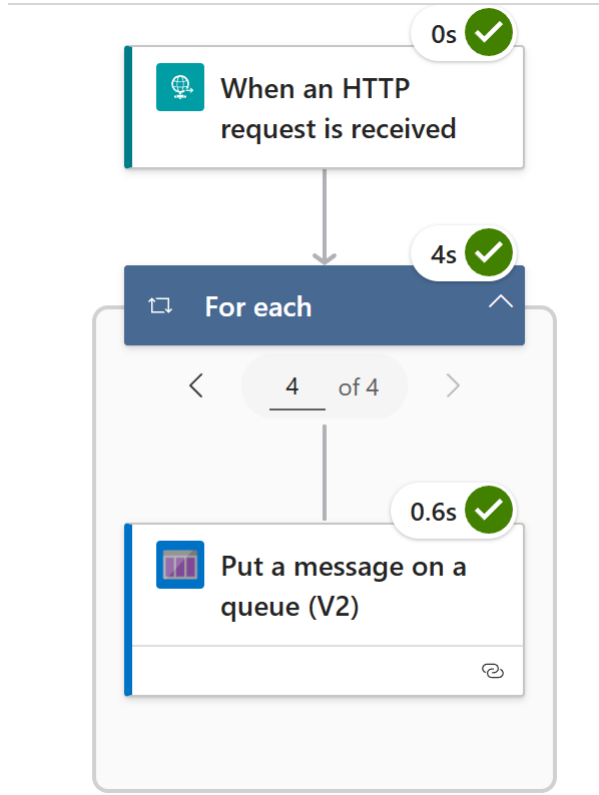


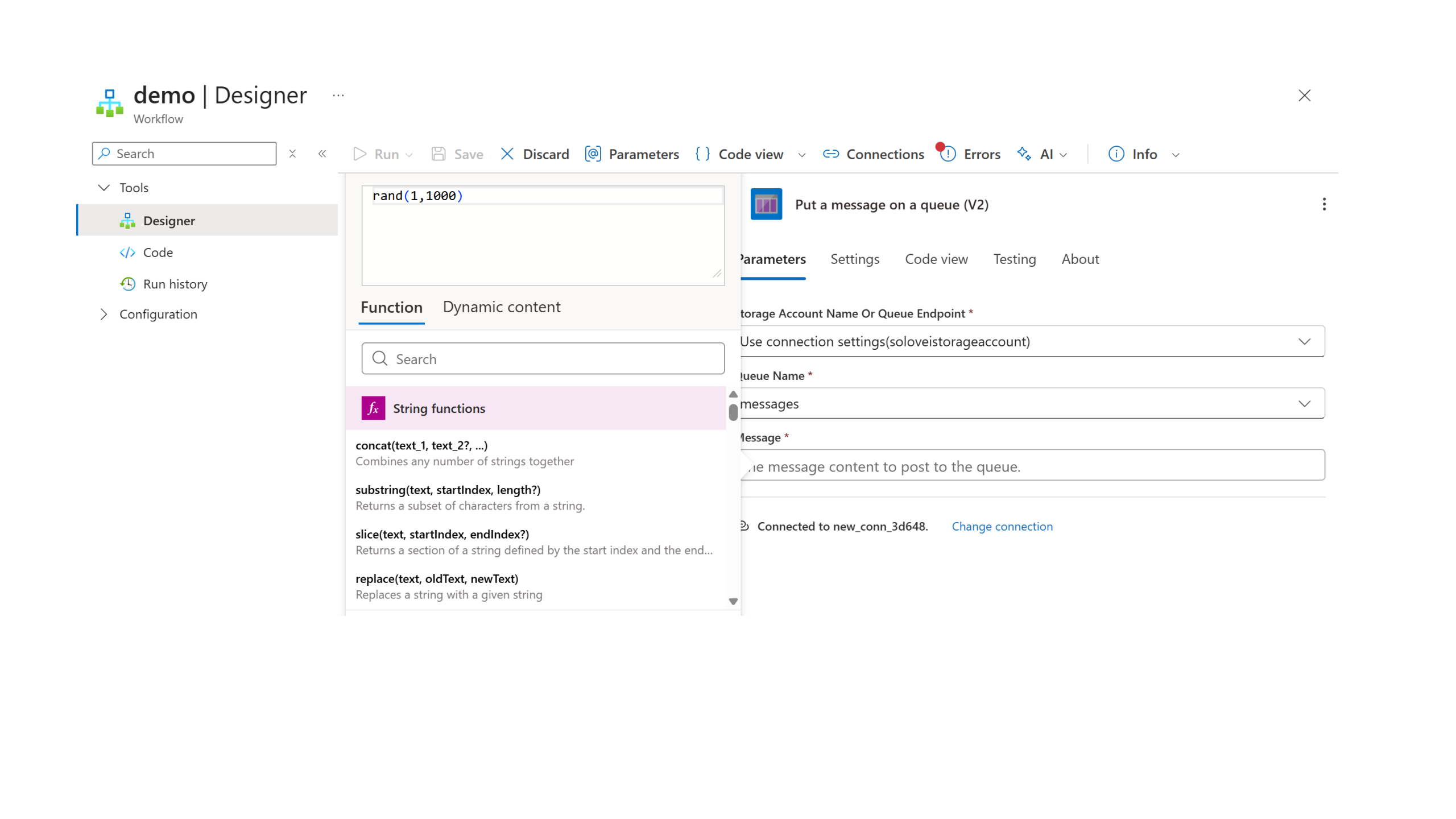
Приклад 1. Зберігти одне повідомлення



Приклад 2. Зберігти повідомлення з повтором

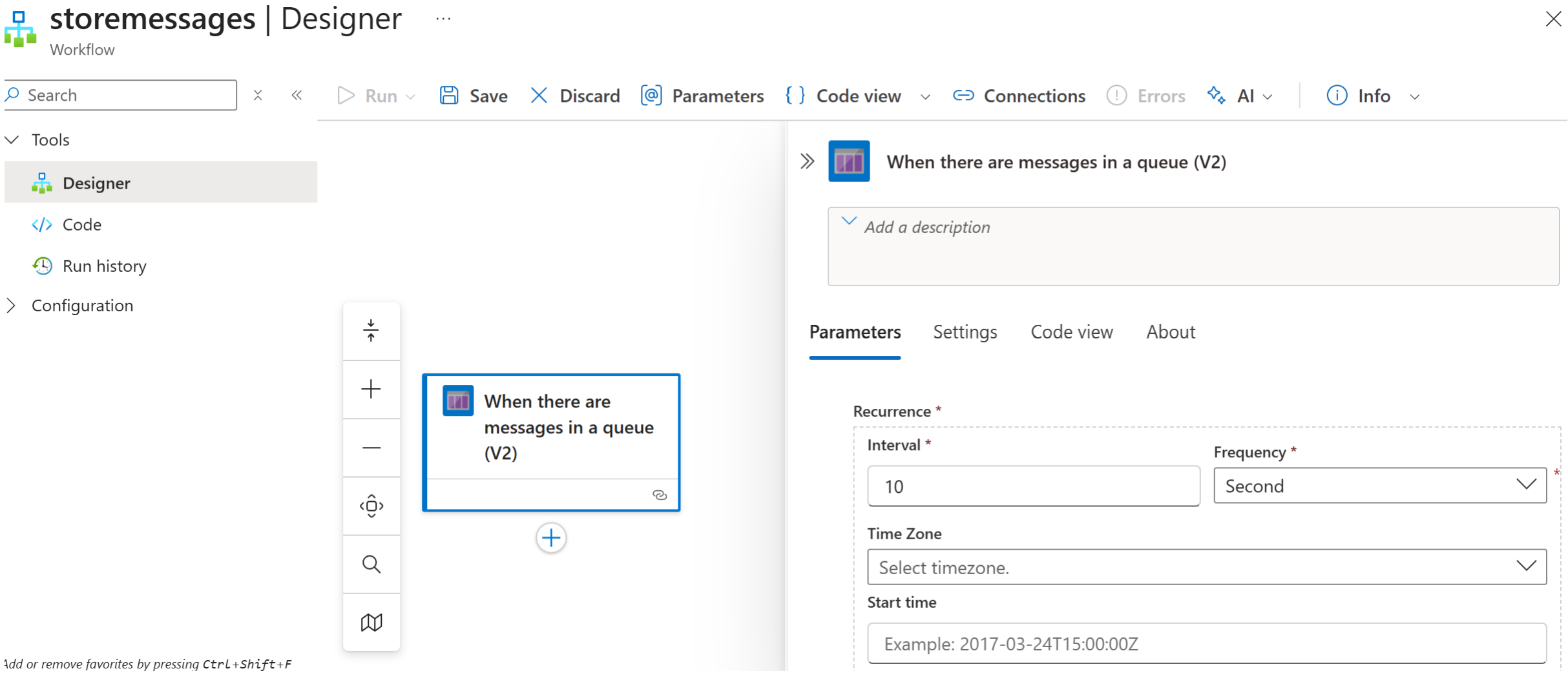





demo | Designer
Workflow
Search
Run
Save
Discard
Parameters
Code view
Connections
Errors
AI
Info
Tools
Designer
Code
Run history
Configuration
rand(1,1000)
Function
Dynamic content
Search
String functions
concat(text_1, text_2?, ...)
Combines any number of strings together
substring(text, startIndex, length?)
Returns a subset of characters from a string.
slice(text, startIndex, endIndex?)
Returns a section of a string defined by the start index and the end...
replace(text, oldText, newText)
Replaces a string with a given string
Put a message on a queue (V2)
Parameters
Settings
Code view
Testing
About
torage Account Name Or Queue Endpoint *
Use connection settings(soloveistorageaccount)
ueue Name *
messages
Message *
ie message content to post to the queue.
Connected to new_conn_3d648.
Change connection

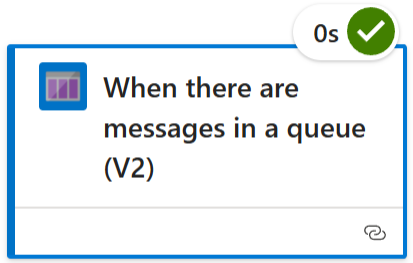
Приклад2. реалізації процесу отримання повідомлення з черзи і збереження

Запускається за допомогою тригера "When there are messages in the queue", щоб додати повідомлення в Azure Blob Storage.



storemessages

Resubmit | Edit | Cancel run | Refresh | Info



When there are messages in a queue (V2)

Add a description

Parameters | Settings | Code view | About

Pop Receipt

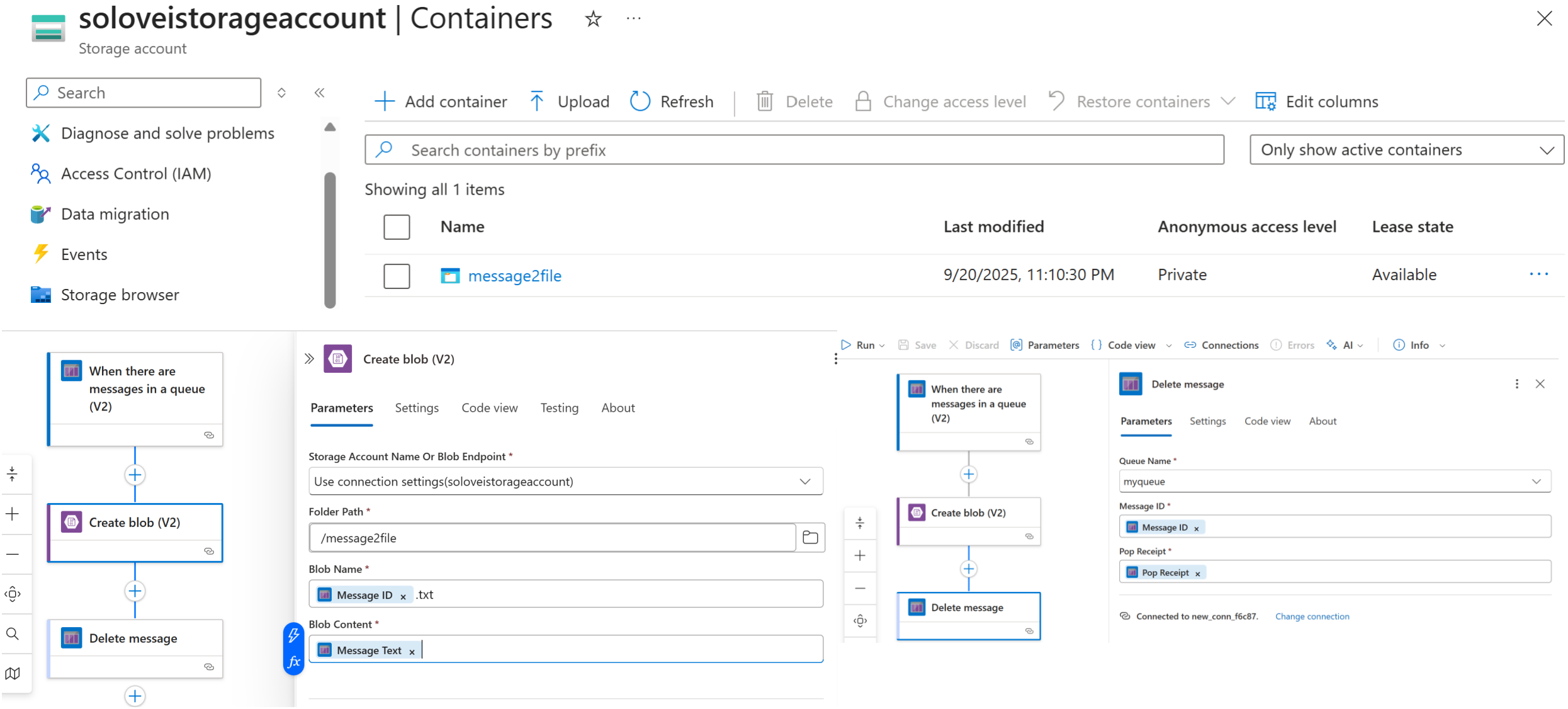
AgAAAAMAAAAAAAAALOg9amYq3AE=

Next Visible Time

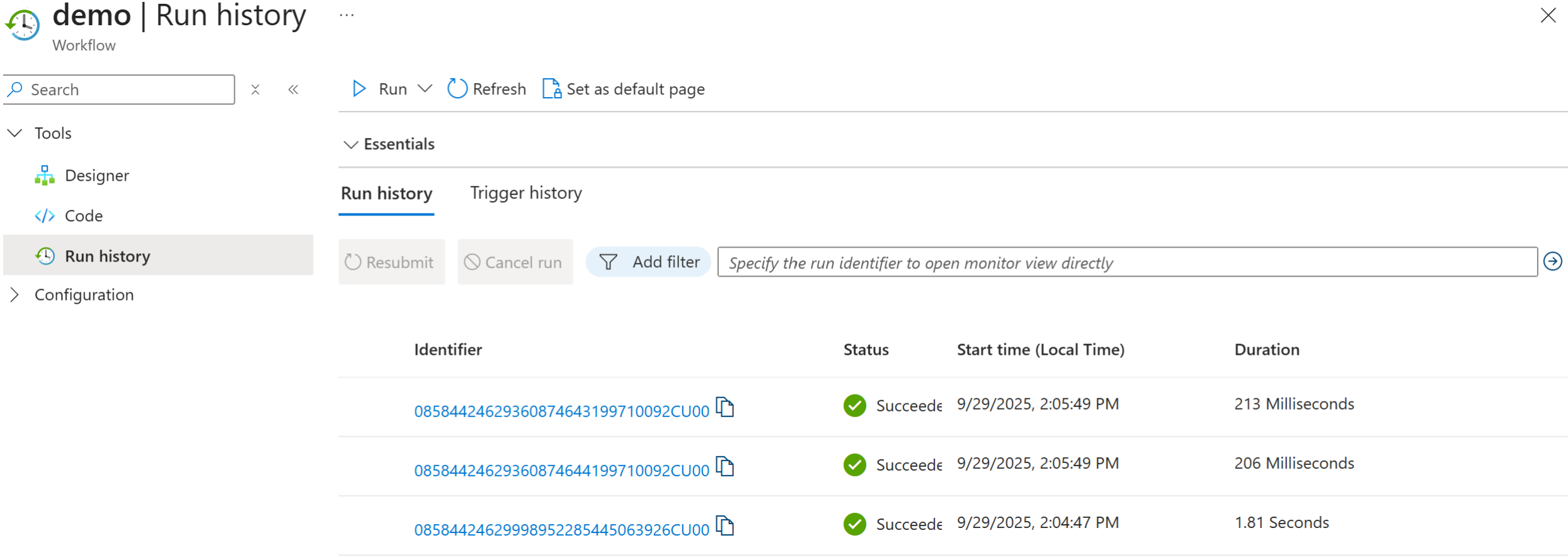
Sat, 20 Sep 2025 19:40:27 GMT

Message Text

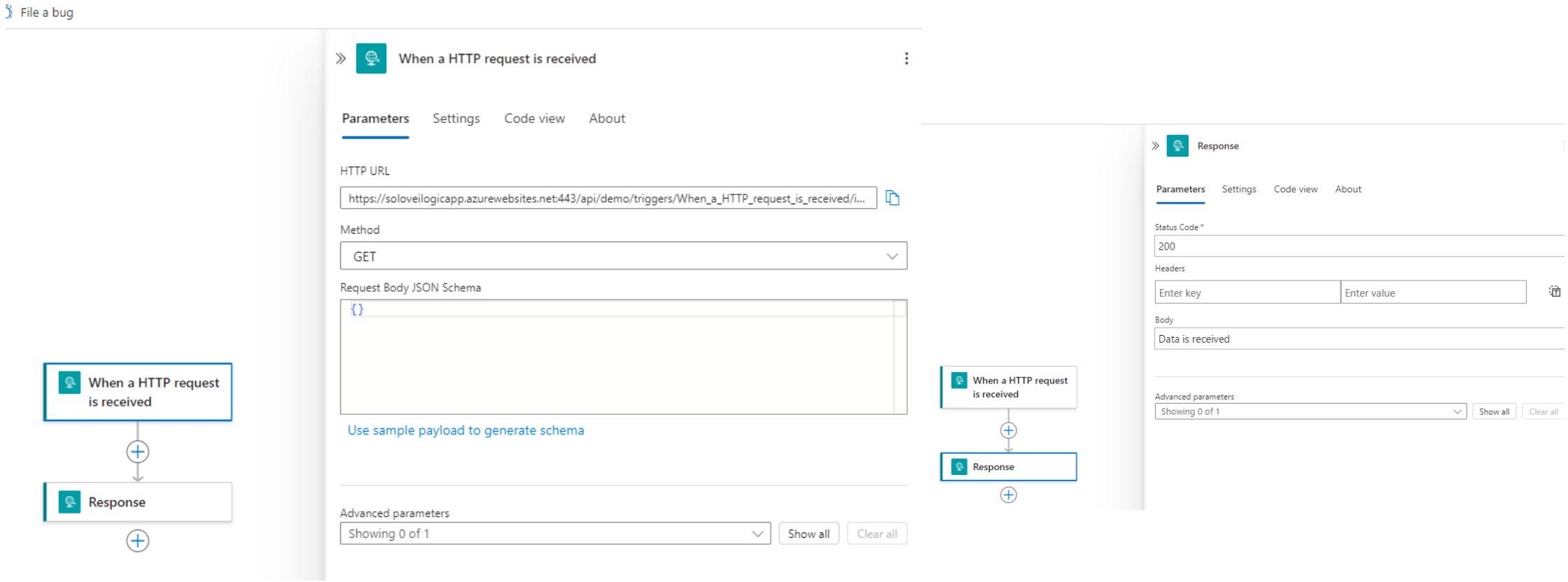
860


soloveistorageaccount | Containers
Storage account
Search
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Add container
Upload
Refresh
Delete
Change access level
Restore containers
Edit columns
Search containers by prefix
Only show active containers
Showing all 1 items
Name
Last modified
Anonymous access level
Lease state
message2file
9/20/2025, 11:10:30 PM
Private
Available
When there are messages in a queue (V2)
Create blob (V2)
Delete message
Create blob (V2)
Parameters
Settings
Code view
Testing
About
Storage Account Name Or Blob Endpoint *
Use connection settings(soloveistorageaccount)
Folder Path *
/message2file
Blob Name *
Message ID
.txt
Blob Content *
Message Text
Run
Save
Discard
Parameters
Code view
Connections
Errors
AI
Info
When there are messages in a queue (V2)
Create blob (V2)
Delete message
Delete message
Parameters
Settings
Code view
About
Queue Name *
myqueue
Message ID *
Message ID
Pop Receipt *
Pop Receipt
Connected to new_conn_f6c87.
Change connection

Історія виконаних завдань



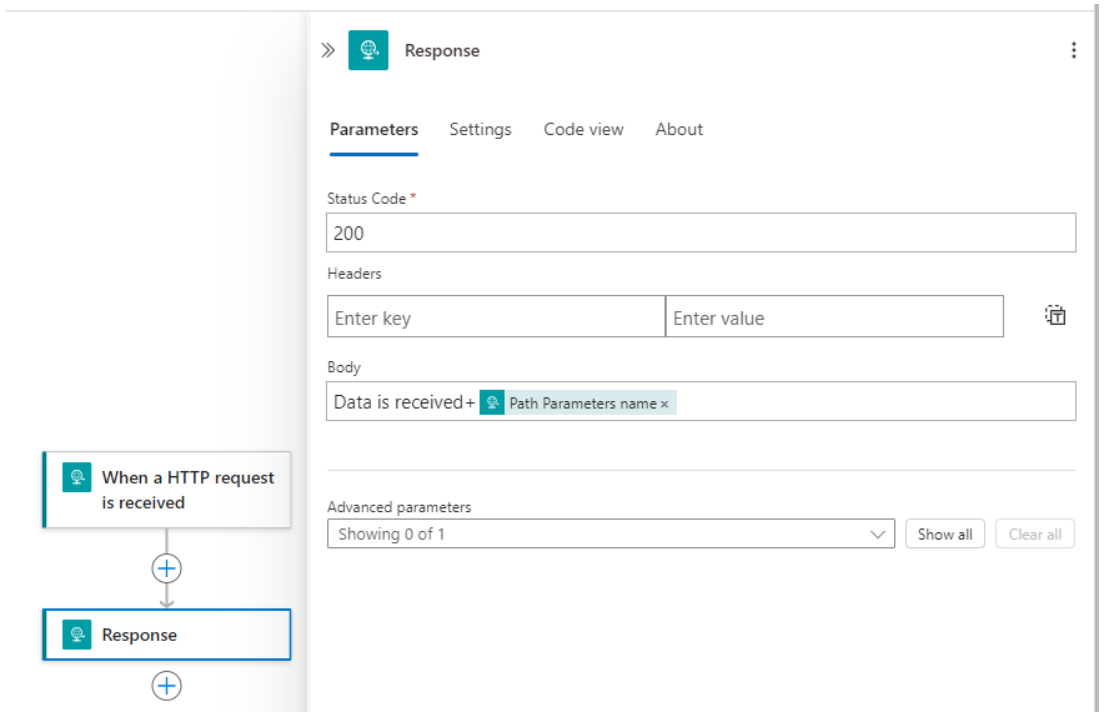
Створюємо Request-Response (без параметрів)

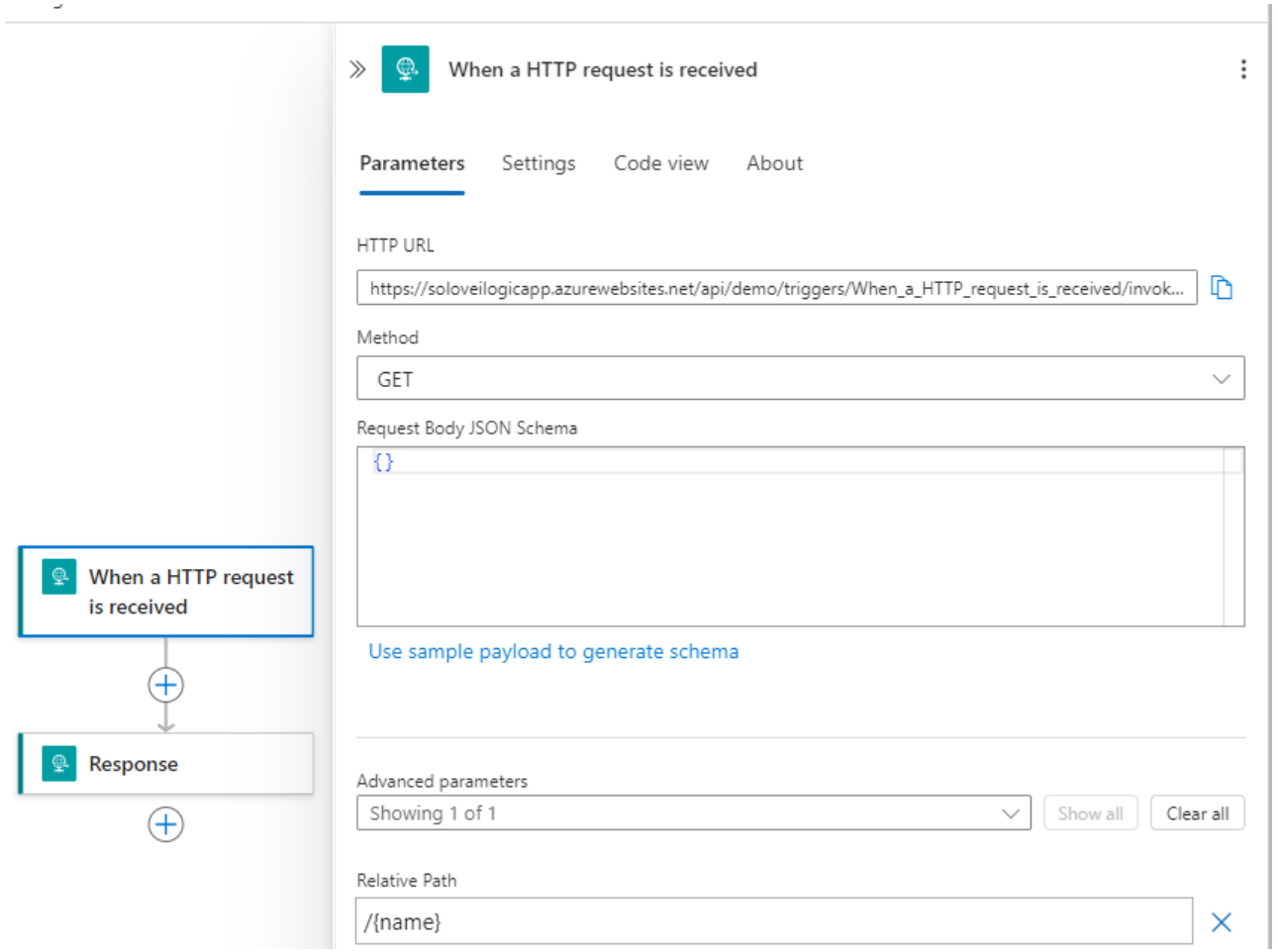


Перевіремо url:
https://soloveilogicapp.azurewebsites.net:443/api/demo/triggers/When_a_HTTP_request_is_received/invoke?api-version=2022-05-01&sp=%2Ftriggers%2FWhen_a_HTTP_request_is_received%2Frun&sv=1.0&sig=i6vACz5QPiT9vS7NZr63S8vHvnl5NflPkOZqudgCjyM

Приклад 3. Створюємо Request-Response (з параметром)

https://soloveilogicapp.azurewebsites.net/api/demo/triggers/When_a_HTTP_request_is_received/invoke/%7Bname%7D?api-version=2022-05-01&sp=%2Ftriggers%2FWhen_a_HTTP_request_is_received%2Frun&sv=1.0&sig=i6vACz5QPiT9vS7NZr63S8vHvnl5NflPkOZqudgCjyM





soloveilogicapp.azurewebsites.net/api/demo/triggers/When_a_HTTP_request_is_received/invoke/%7BXXXXXX%7D?api-version=2022-05-01&sp=%2Ftriggers%2FWhen_a_HTTP_request_is_received%2Frun&sv=1.0&sig=i6vACz5QPiT9vS7NZ...

```
Data is received+{XXXXXX}
```

REST API basics: CRUD, test & variable / **Get data**

Save | Share

GET | https://soloveilogicapp.azurewebsites.net/api/demo/triggers/When_a_HTTP_request_is_received/invoke/%7Bname%7D?api-version=2022-05-01&sp=%2Ftriggers%2FWhen_a_HTTP_request_is_receiv... | Send

Params • | Authorization | Headers (5) | Body | Scripts • | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

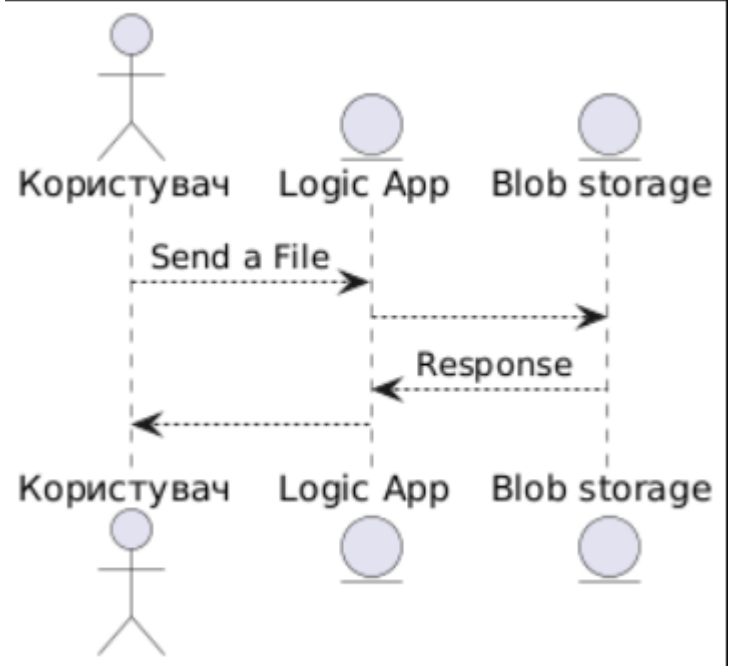
This request does not have a body

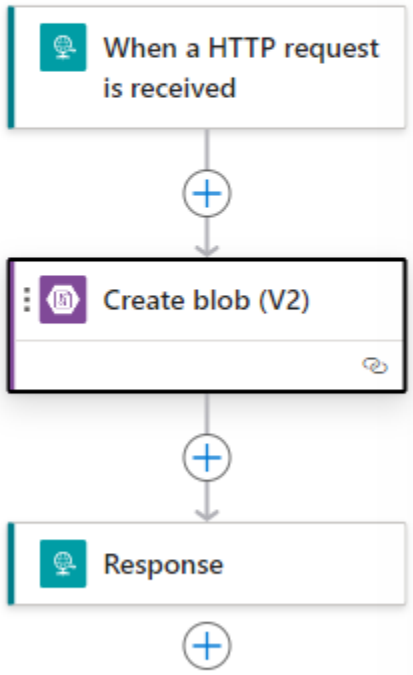
Body | Cookies | Headers (15) | Test Results (1/1)

200 OK • 1290 ms • 757 B | Save Response

Pretty | Raw | Preview | Visualize | Text

```
1  Data is received+{name}
```





Create blob (V2)

Parameters | Settings | Code view | Testing | About

Storage Account Name Or Blob Endpoint *

Use connection settings(soloveistoragedemo)

Folder Path *

/demo

Blob Name *

Path Parameters name × .txt

Blob Content *

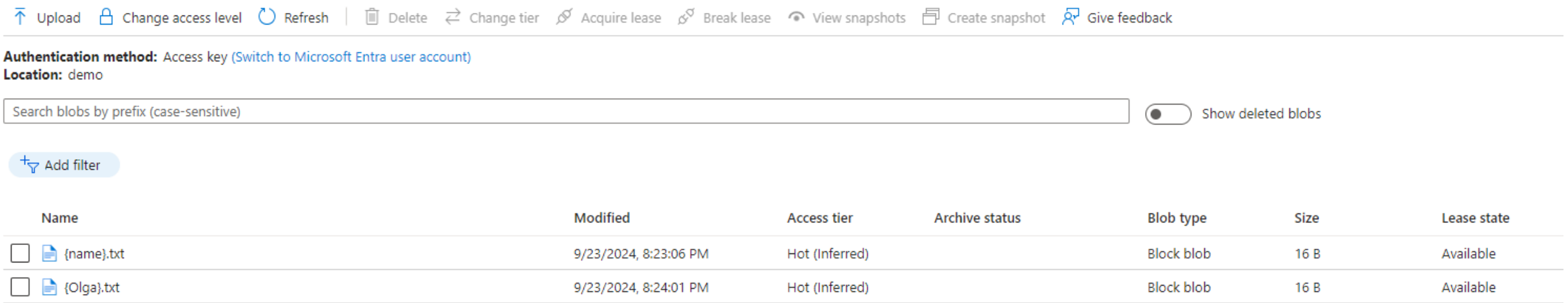
Data from Path Parameters name ×

Advanced parameters

Showing 0 of 1 | Show all | Clear all

Connected to simple. Change connection

Результат – доданий файл



Use the Azure Queue Storage client library for Python to: ***pip install azure-storage-queue***

1. Create a queue
2. Add messages to a queue
3. Peek at messages in a queue
4. Update a message in a queue
5. Get the queue length
6. Receive messages from a queue
7. Delete messages from a queue
8. Delete a queue

QueueClient - Представляє конкретну чергу. Дозволяє: Створювати / видаляти чергу. Додавати, отримувати, видаляти повідомлення.	queue_client = queue_service_client.get_queue_client("my-queue")
Створення черги	queue_client.create_queue()
Видалення черги	queue_client.delete_queue()
Перевірка наявності черги	exists = queue_client.exists()
Додавання повідомлення до черги	queue_client.send_message("Привіт з Azure Queue!")
Отримання повідомлень Параметри: max_messages: кількість повідомлень. visibility_timeout: час, на який приховати повідомлення.	messages = queue_client.receive_messages() for message in messages: print(message.content)

Видалення повідомлення	queue_client.delete_message(message)
Повідомлення з обмеженням часу життя	queue_client.send_message("Тимчасове повідомлення", time_to_live=3600) # 1 година
Затримка перед появою в черзі	queue_client.send_message("Затримане повідомлення", visibility_timeout=60)
Отримання кількості повідомлень у черзі	props = queue_client.get_queue_properties() print(props.approximate_message_count)
Додавання метаданих	queue_client.set_metadata({'project': 'AI', 'env': 'test'})
Робота з пакетами повідомлень (Batch) Azure Queue підтримує обробку до 32 повідомлень одночасно. Потрібно використовувати .receive_messages(max_messages=32).	

Приклад роботи з функціями бібліотеки ***azure-storage-queue***

```
import time
import uuid
from azure.storage.queue import QueueServiceClient
from azure.storage.blob import BlobServiceClient

# Налаштування
STORAGE_CONNECTION_STRING = "<YOUR_STORAGE_CONNECTION_STRING>"
QUEUE_NAME = "my-queue"
CONTAINER_NAME = "mycontainer"
CHECK_INTERVAL_SECONDS = 10  # Перевіряти кожні 10 секунд

# Підключення до сервісів
queue_service = QueueServiceClient.from_connection_string(STORAGE_CONNECTION_STRING)
blob_service = BlobServiceClient.from_connection_string(STORAGE_CONNECTION_STRING)

queue_client = queue_service.get_queue_client(QUEUE_NAME)
container_client = blob_service.get_container_client(CONTAINER_NAME)


while True:
    print("Перевірка черги на наявність повідомлень...")

    messages = queue_client.receive_messages(messages_per_page=1, visibility_timeout=30)

    for msg_batch in messages.by_page():
        for msg in msg_batch:
            message_text = msg.content
            print(f"Отримано повідомлення: {message_text}")

            # Створення унікального імені для Blob-файлу
            blob_name = f"message-{uuid.uuid4()}.txt"
            blob_client = container_client.get_blob_client(blob_name)

            # Завантаження повідомлення у Blob Storage
            blob_client.upload_blob(message_text)
            print(f"Збережено в Blob Storage як '{blob_name}'")

            # Видалення повідомлення з черги після обробки
            queue_client.delete_message(msg)
            print("Повідомлення видалено з черги")

    print(f"Очікування {CHECK_INTERVAL_SECONDS} секунд...\n")
    time.sleep(CHECK_INTERVAL_SECONDS)
```

AS (Shared Access Signature)

AS (Shared Access Signature) — це рядок (токен), який дозволяє обмежений доступ до ресурсів Azure Storage (Blob, Queue, Table, File), без необхідності розкривати основні ключі облікового запису.
За допомогою SAS можна контролювати:

- ✓ які ресурси доступні,
- ✓ які права (read, write, add, delete тощо),
- ✓ з якого часу по який (тобто TTL),
- ✓ через який протокол (HTTP / HTTPS),
- ✓ з якої IP-адреси або діапазону IP.

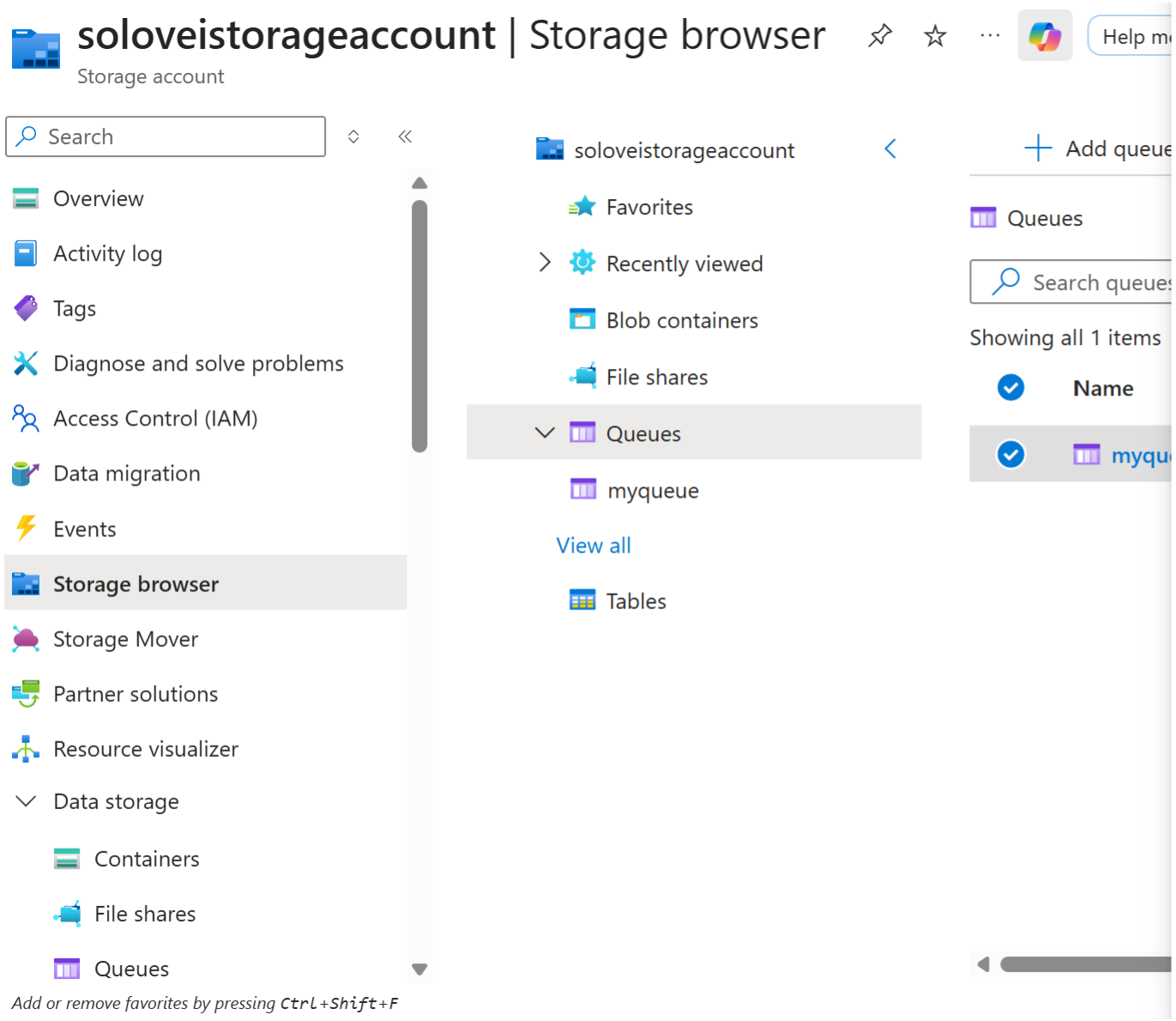
Типи SAS

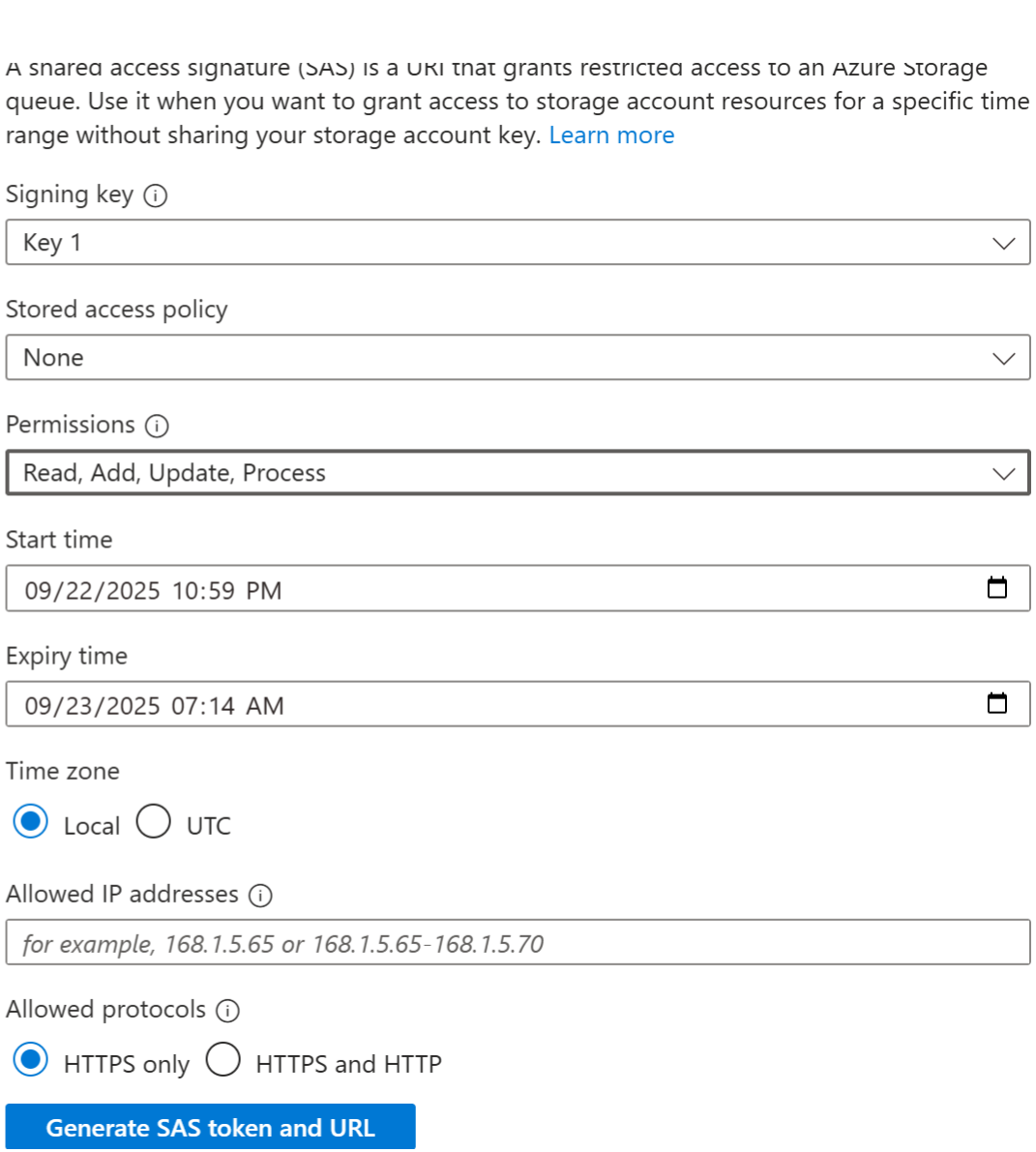
Service SAS — дозволяє доступ до ресурсів одного з сервісів (Blob, Queue, Table, Files) за допомогою ключа облікового запису.
Account SAS — дає доступ до декількох сервісів (Blob + Queue + Table + File) в обліковому записі.
User Delegation SAS — підписується за допомогою делегованого ключа користувача через Azure AD. Але Queue сервіс не підтримує User Delegation SAS на даний момент (для черг лише обліковий ключ чи сервісний SAS).

Основні параметри SAS Token — що вони значать

Параметр	Позначення	Що задає / обмежує
sv	*signed version*	Версія API сервісу Storage, для якої дійсний SAS. Напр., дата, коли версія змінена.
ss	*signed services*	Для Account SAS — які сервіси охоплює токен (Blob b, Queue q, Table t, File f).
srt	*signed resource types*	Типи ресурсів, до яких дозволено доступ: «сервіс» (service), «контейнер/черга» (container / queue), «об’єкт» (object/message).
sp	*signed permissions*	Що можна робити: читати, писати, видаляти тощо. Для черг це можуть бути такі права, як add, read, process тощо.
se	*signed expiry*	коли SAS закінчується — дата й час в UTC.
st	*signed start*	з якого часу SAS стає дійсним. Опційний, але корисний, щоб не можна було використати до певного моменту.
spr	*signed protocol*	Який протокол дозволений: https лише чи http,https. Для безпеки часто https.
sig	*signature*	Підпис (HMAC-SHA256), який створюється з наведених вище параметрів + секретного ключа облікового запису. Azure перевіряє цей підпис, щоб підтвердити, що токен справжній і не змінений.

Створити SAS ключ для облікового запису сховища





До отриманого ключа – додати назву черги + «messages»

https://soloveistorageaccount.queue.core.windows.net/myqueue/messages?sv=2024-11-04&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2025-09-29T17:44:57Z&st=2025-09-29T09:29:57Z&spr=https,http&sig=QR2G6UT%2BOKfTFN2%2FoInpiD%2B%2BIoR5MGQOFLDNxCBbQuA%3D

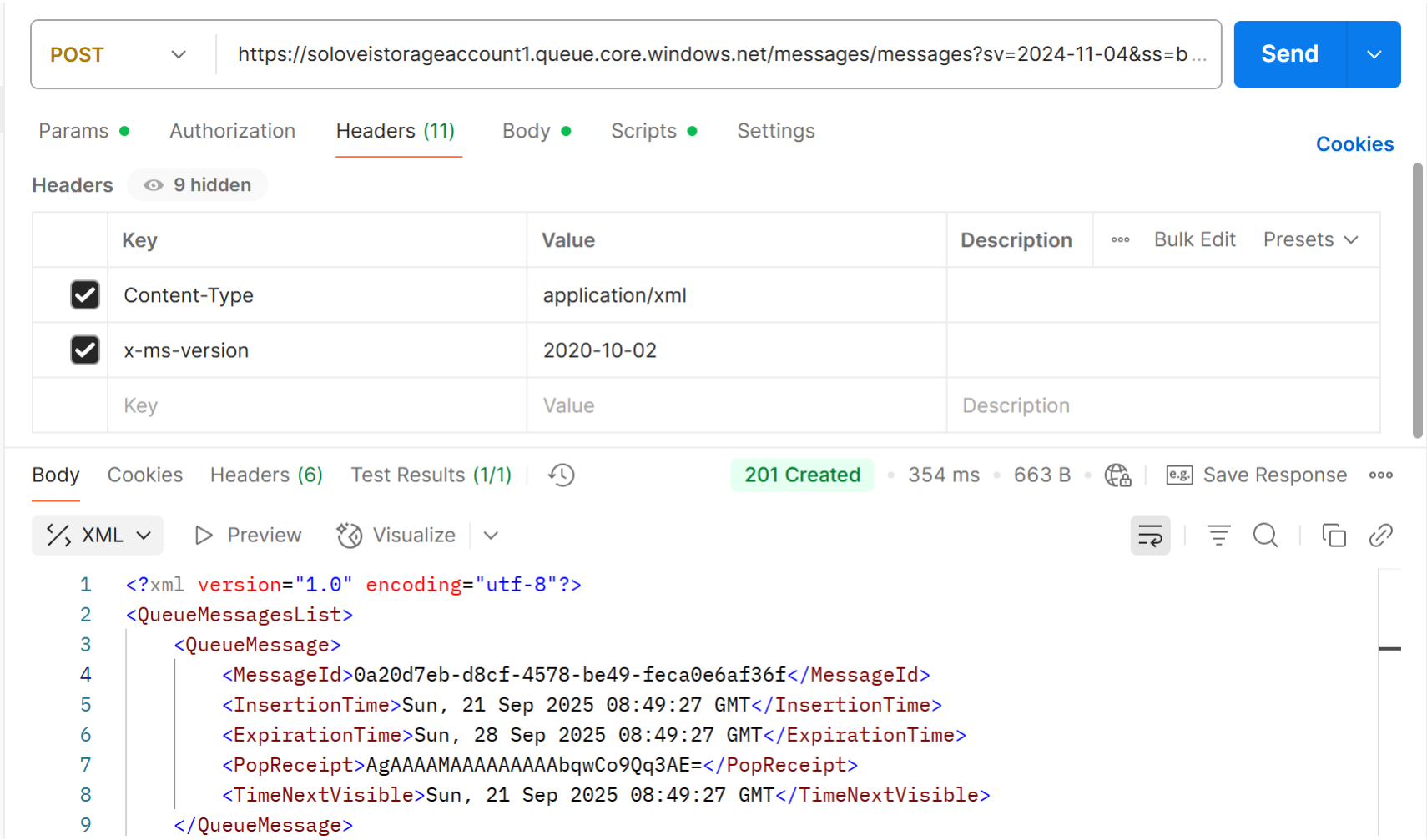
Приклад SAS URL

Частина	Що означає
https://soloveistorageaccount1.queue.core.windows.net/messages/messages?	Це базова URL-адреса ресурсу. Має вказувати на конкретну чергу і endpoint, до якого звертаємось (/messages). У вашому прикладі /messages/messages виглядає дивно — зазвичай буває <storage_account>.queue.core.windows.net/<queue_name>/messages. Ім'я черги — воно перше /messages після домену.
sv=2024-11-04	Версія сервісу Azure Storage, яка використовується. Вона визначає, які функціональності доступні, які правила форматування, обробки.
ss=bfqt	Сервіси, до яких дається доступ у SAS. b = Blob, f = File, q = Queue, t = Table. Оскільки bfqt — означає доступ до усіх цих сервісів. Однак якщо ви робите SAS тільки для черги, можна було б обмежити тільки q.
srt=sco	Типи ресурсів: s = Service-level, c = Container / Queue-level (те, що дозволяє працювати з чергою як цілим контейнером / чергою), o = об'єкт (наприклад, окреме повідомлення). Тут sco — доступ до сервісного рівня, черги/контейнера і об'єктів.
sp=rwdlacupiytfx	Права, які даються SAS. Це комбінація літер, кожна означає різне право, для прикладу: r = read, w = write, d = delete, l = list, a = add, c = create, u = update, p = process, i, y, t, f, x — залежно від сервісу і версії. Тут виглядає як дуже розширені права. Це означає, що з токеном можна буквально багато операцій зробити. (У продакшн зазвичай дають мінімально необхідні права.)
se=2025-09-21T16:36:58Z	Дата та час, до якого SAS діє (закінчується) (в UTC). Після цього використовувати SAS не можна.
st=2025-09-21T08:21:58Z	Початок дії SAS. До цього часу токен може бути недійсним, залежно від сервісу/налаштувань. Це корисно, щоб уникнути одразу використання токена в минулому або кешованих запитів.
spr=https	Протокол, який дозволений. Тут — тільки HTTPS. Це добре з міркувань безпеки.
sig=…	Підпис. Це HMAC підпис, який обчислюється на основі всієї сукупності параметрів SAS (версія, права, строки, ресурс і т.д.) з використанням ключа облікового запису. Azure при запиті перевіряє підпис і якщо все співпадає — дозволяє операцію.

Header+ Додаємо SAS ключ як посилання в метод POST

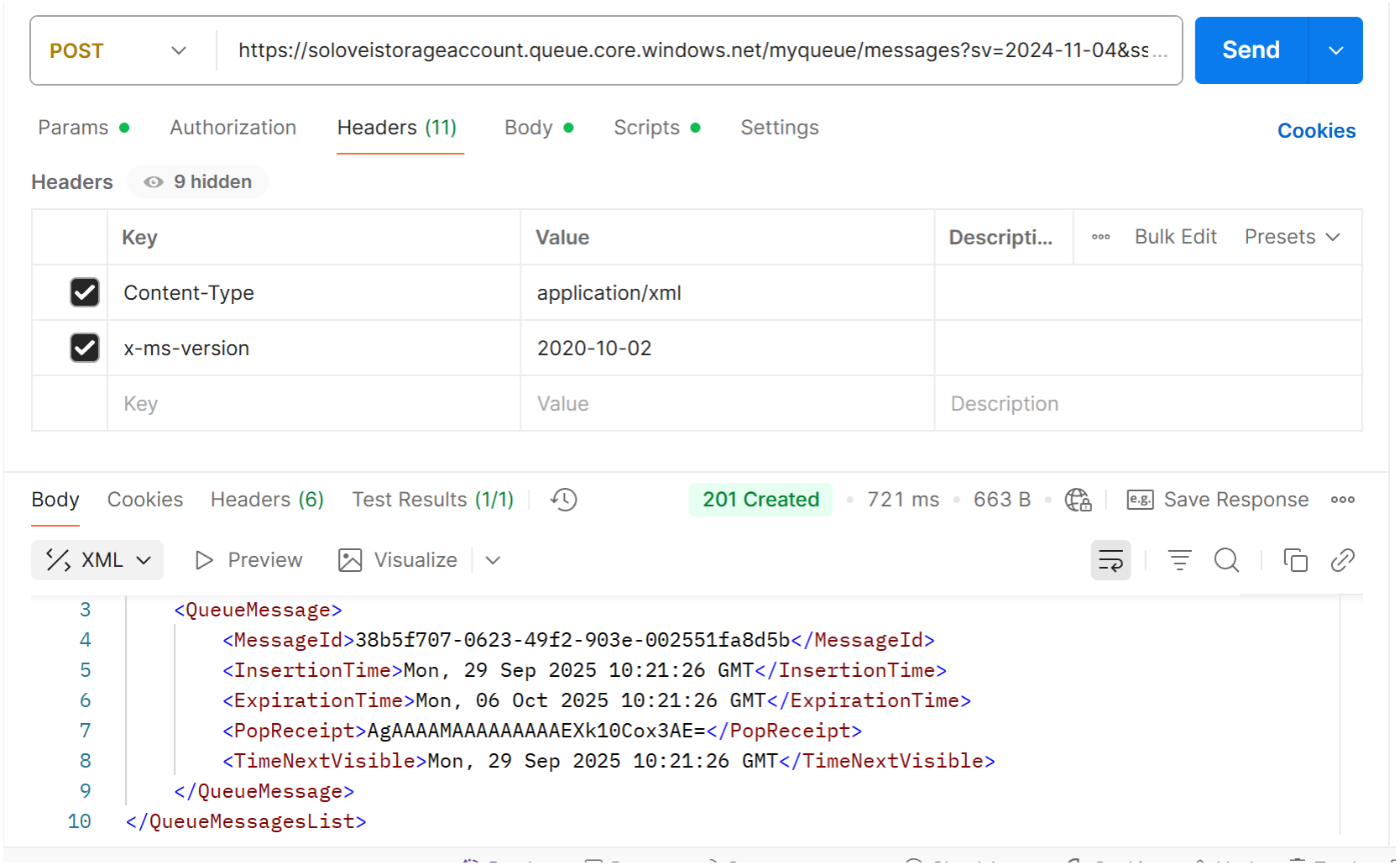
Content-Type – application/xml

x-ms-version = 2020-10-02



Додаємо SAS ключ як посилання в метод POST

<QueueMessage>
<MessageText>Test message</MessageText >
</QueueMessage>



Питання

1. Що таке Storage Account в Azure і яку роль він відіграє в роботі з Queue Storage?
2. Які основні характеристики Azure Queue (розмір повідомлення, кількість черг, типи черг)?
3. Яку структуру має Message у Queue Storage та які обмеження накладаються на його вміст і розмір?
4. Як працює механізм visibility timeout після отримання повідомлення з черги?
5. У чому різниця між Peek та Dequeue операціями в Azure Queue Storage?
6. Які переваги використання Azure Logic Apps для обробки повідомлень з Azure Queue Storage?
7. Який тригер використовується в Logic Apps для автоматичного зчитування нових повідомлень з черги?
8. Як можна реалізувати обробку повідомлень у циклі в Logic Apps при великому потоці даних?
9. Що відбувається з повідомленням після того, як Logic App успішно його опрацює?
10. Як налаштувати error handling у Logic Apps при виникненні помилки під час обробки повідомлення з черги?
11. Яку бібліотеку Python використовується для роботи з Azure Queue Storage, та як її встановити?
12. Як створити клієнт для черги за допомогою QueueServiceClient або QueueClient?
13. Яким способом у Python SDK можна додати повідомлення до черги й які параметри можна передати?
14. Як отримати та видалити повідомлення з черги за допомогою Python SDK?
15. Яким чином можна оновити існуюче повідомлення в Azure Queue Storage через Python?