

한국 마이크로소프트

Microsoft Technical Trainer

Enterprise Skills Initiative

Microsoft Azure

Azure Event Hubs

이 문서는 Microsoft Technical Trainer팀에서 ESI 교육 참석자분들에게 제공해 드리는 문서입니다.

요약

이 내용들은 표시된 날짜에 Microsoft에서 검토된 내용을 바탕으로 하고 있습니다. 따라서, 표기된 날짜 이후에 시장의 요구사항에 따라 달라질 수 있습니다. 이 문서는 고객에 대한 표기된 날짜 이후에 변화가 없다는 것을 보증하지 않습니다.

이 문서는 정보 제공을 목적으로 하며 어떠한 보증을 하지는 않습니다.

저작권에 관련된 법률을 준수하는 것은 고객의 역할이며, 이 문서를 마이크로소프트의 사전 동의 없이 어떤 형태(전자 문서, 물리적인 형태 막론하고) 어떠한 목적으로 재 생산, 저장 및 다시 전달하는 것은 허용되지 않습니다.

마이크로소프트는 이 문서에 들어있는 특허권, 상표, 저작권, 지적 재산권을 가집니다. 문서를 통해 명시적으로 허가된 경우가 아니면, 어떠한 경우에도 특허권, 상표, 저작권 및 지적 재산권은 다른 사용자에게 허여되지 않습니다.

© 2022 Microsoft Corporation All right reserved.

Microsoft®는 미합중국 및 여러 나라에 등록된 상표입니다.

이 문서에 기재된 실제 회사 이름 및 제품 이름은 각 소유자의 상표일 수 있습니다.

문서 작성 연혁

1. 이 문서는 2022년 01월 31일 한국 마이크로소프트(유), Microsoft Technical Trainer 팀 우진환에 의해 TASK 01 ~ TASK 02부분이 작성되었습니다. (Version 0.4.0)
2. 이 문서는 2022년 02월 02일 한국 마이크로소프트(유), Microsoft Technical Trainer 팀 우진환에 의해 TASK 03 ~ TASK 04 부분이 추가되었습니다. (Version 0.8.0)
3. 이 문서는 2022년 02월 03일 한국 마이크로소프트(유), Microsoft Technical Trainer 팀 우진환에 의해 TASK 05 부분이 추가되었습니다. (Version 1.0.0)

목차

기본 개념.....	5
파티셔닝(PARTITIONING).....	5
네임스페이스(NAMESPACES)	6
이벤트 소비자(EVENT CONSUMER)	7
TASK 01. EVENT HUB 만들기.....	9
TASK 02. 데이터 출력 애플리케이션 만들기.....	13
TASK 03. LOGIC APPS를 통해 EVENT HUB로 이벤트 전송	21
TASK 04. EVENT HUB 수신 애플리케이션 만들기.....	28
TASK 05. 이벤트 캡처.....	39

Azure Event Hubs는 Azure의 빅 데이터 메시징 서비스이며 스트리밍 플랫폼 및 이벤트 수집 서비스입니다. Azure Event Hubs를 통해 초당 수백만개의 이벤트를 수신하고 처리할 수 있으며 Event Hubs로 전송된 데이터는 실시간 분석 공급자 또는 일괄 처리/스토리지 어댑터를 사용하여 변환하고 저장할 수 있습니다. 다음과 같은 시나리오에서 Event Hubs를 사용할 수 있습니다.

- 이상 행위 탐지(사기 행위/비정상적인 값)
- 애플리케이션 로깅
- 클릭스트림과 같은 분석 파이프라인
- 라이브 대시보드
- 데이터 아카이빙
- 트랜잭션 처리
- 사용자 원격 분석 처리
- 디바이스 원격 분석 스트리밍

Azure Event Hub는 다음과 같은 서비스 특징을 가지고 있습니다.

- 빅 데이터 스트리밍 서비스이기 때문에 서비스에 데이터를 지속적으로 보내는 프로세스라고 가정합니다.
- terabyte 데이터 크기를 처리할 수 있으며 초당 수 백만개의 이벤트로 확장할 수 있습니다.
- 오류 불가지론(agnostic to failures)으로 디자인되었기 때문에 데이터 손실이 없어 안정적입니다.
- Event Hub는 여러 프로토콜과 SDK를 지원합니다. 표준 HTTP, AMQP 프로토콜을 사용할 수 있습니다.

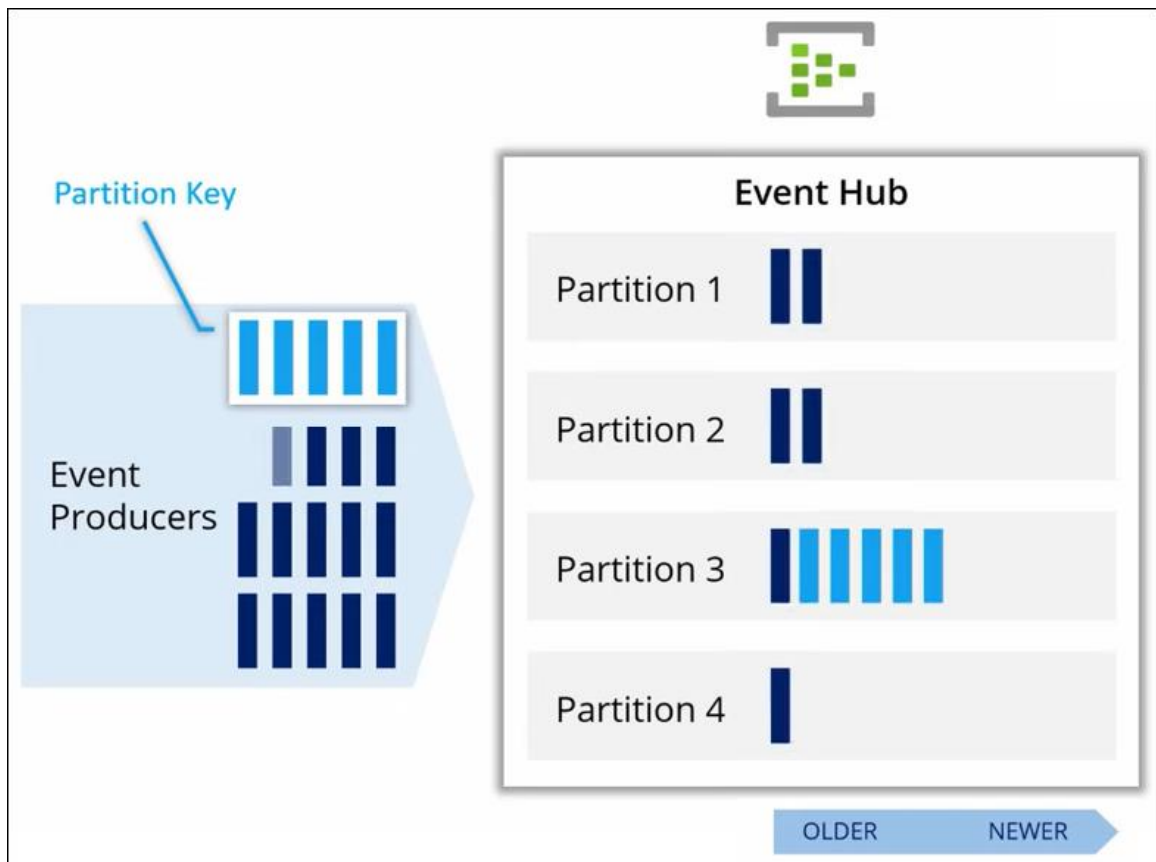
기본 개념

Event Hub로 데이터를 보내는 모든 엔터티는 이벤트 게시자(Event Publisher)이며 Event Producer라고도 합니다. Event Publisher는 HTTPS, AMQP 1.0 (Advanced Messaging Queuing Protocol), Kafka 프로토콜을 사용하여 이벤트를 게시할 수 있습니다. 이러한 서비스는 이벤트의 소스이며 이러한 이벤트가 Event Hub로 전송됩니다.

파티셔닝(Partitioning)

- Event Hub는 Event Hub로 전송되는 이벤트의 시퀀스를 하나 이상의 파티션으로 구성합니다.
- Event Hub를 생성할 때 파티션 수를 선택하고 생성한 후에는 변경할 수 없기 때문에 어떤 종류의 파티셔닝과 몇 개의 파티션이 필요한지 미리 선택하는 것이 좋습니다. 잘못 구성한 경우 Event Hub를 다시 생성해야 합니다.
- 메시지를 보내기 시작하면 메시지는 이러한 파티션 간 로드 밸런싱됩니다. 파티션으로 로드 밸런싱이 되지만 파티션이 동일하게 활용된다는 보장은 없기 때문에 각 파티션이 다른 식으로 커질 것이라고 예상해야 합니다.
- 각 파티션은 파티션 내에서 이벤트를 오래된 것부터 최신 순서로 정렬합니다. 이러한 방식은 큐(queue)와 유사하지만 이벤트가 여러 파티션으로 로드 밸런싱되기 때문에 로드 밸런싱된 이벤트가 여러 파티션에서 순서대로 처리된다고 보면 안 됩니다.
- 여러 이벤트를 순서대로 처리해야 하는 경우 업로드하는 이벤트에 대한 파티션 키(partition key)를 사용해야 합니다. 이벤트에 파티션 키를 지정하면 이 이벤트는 모두 동일한 파티션에서 처리됩니다. 이를 통해 이벤트가

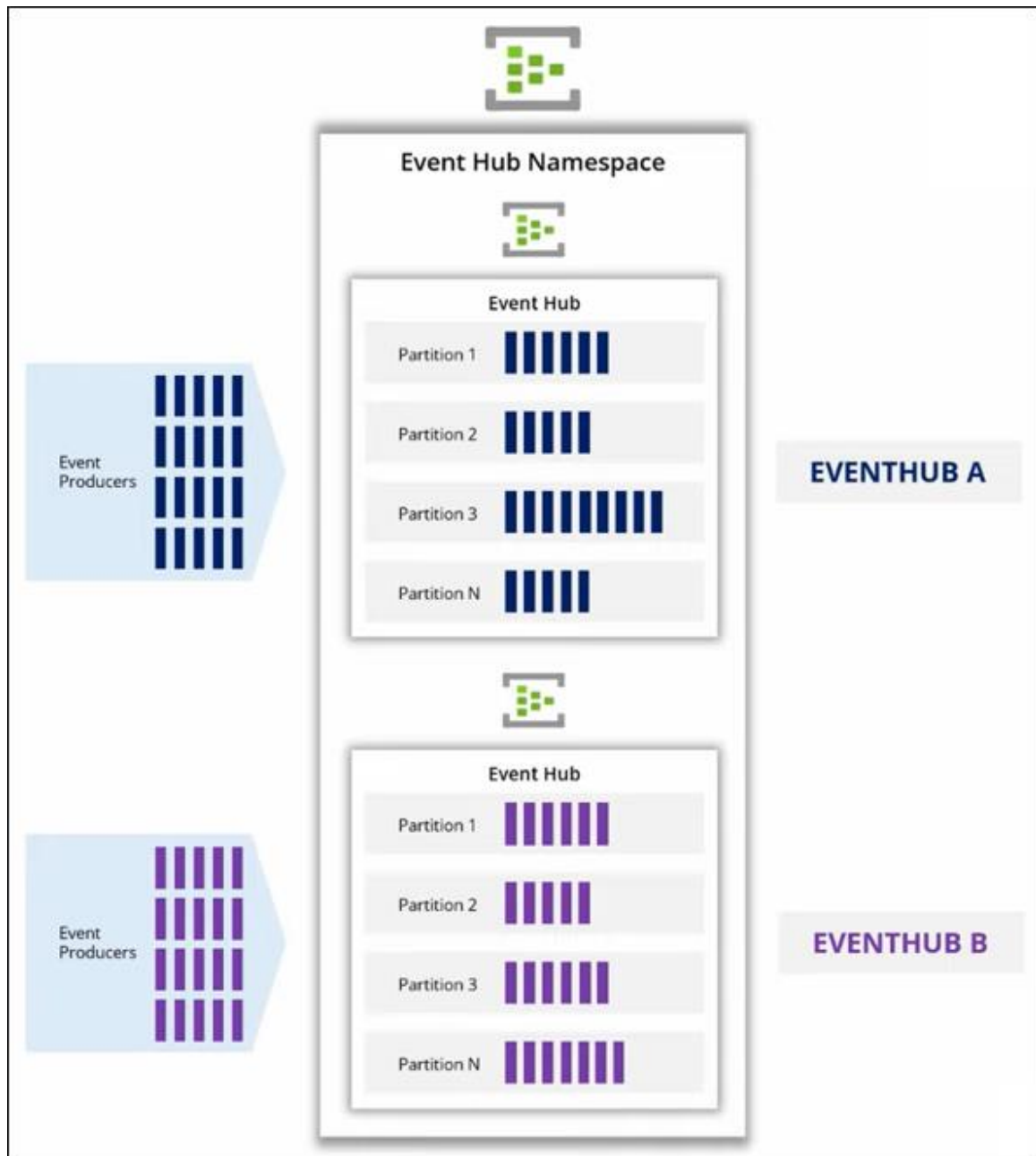
동일한 파티션에 전송되고 순서대로 처리되는지 확인할 수 있습니다.



네임스페이스(Namespace)

Event Hubs 네임스페이스는 Event Hub에 대한 관리 컨테이너입니다. 예를 들어 고유한 데이터 스트림을 나타내는 Event Hub A가 있고 다른 데이터를 포함하는 두 번째 고유한 데이터 스트림을 처리하는 Event Hub B가 있는 경우, 여러 Event Hub를 만들기 위한 논리 컨테이너를 네임스페이스라고 합니다. 네임스페이스는 공유 처리량, 공유 비용 등과 같은 여러 공유 속성을 가지고 있는 범위를 지정하는 컨테이너입니다.

- Event Hub는 고유한 데이터 스트림을 표현합니다.
- Event Hub 네임스페이스는 Event Hub의 컬렉션이며 컨테이너의 범위를 지정하고 공유 속성을 공유하며 FQDN을 통해 액세스할 수 있습니다.



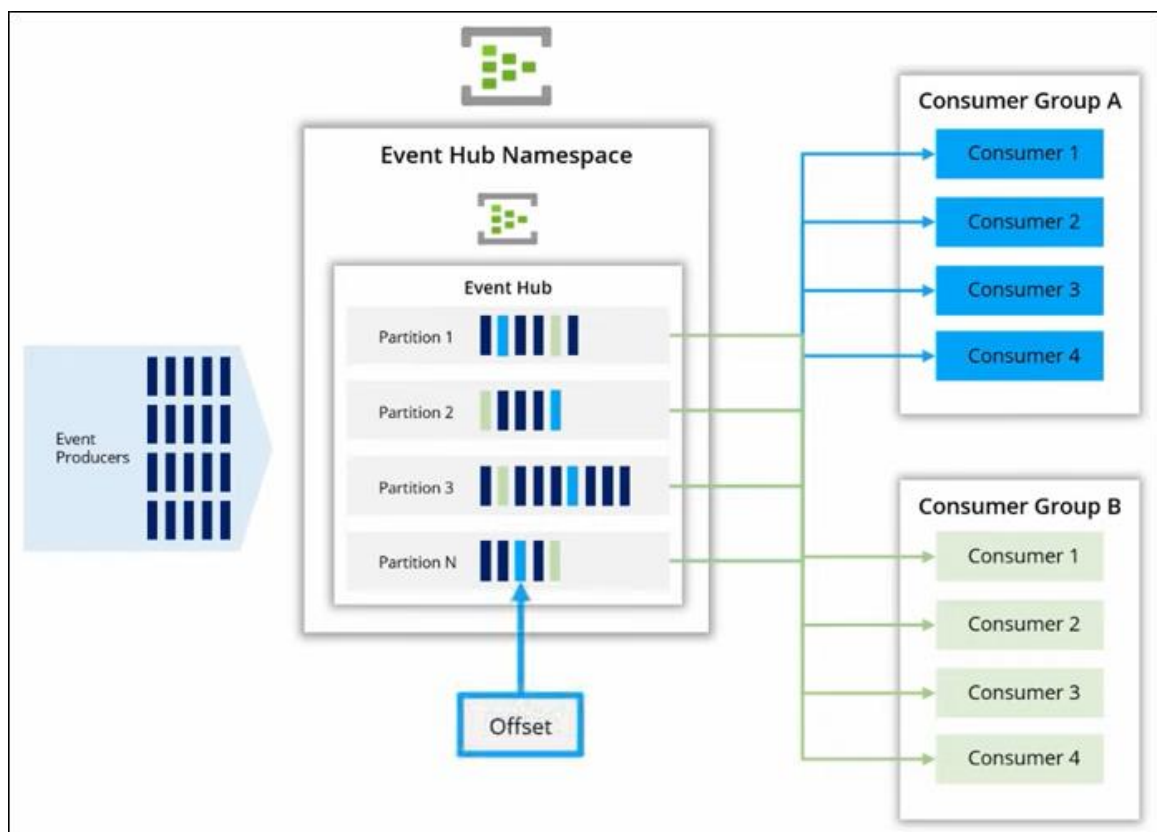
이벤트 소비자(Event Consumer)

Consumer Group은 가격 책정에 따라 선택할 수 있으며 Event Hub에서 이벤트 데이터를 읽는 모든 엔터티를 말합니다. 따라서 Consumer Group을 Event Hub 데이터에 대한 고유한 보기(view)로 생각할 수 있습니다. 즉 각 Consumer Group은 전체 Event Hub 데이터에서 개별적으로 읽을 수 있는 Event Hub 데이터에 대해 개별적인 보기를 가질 수 있다는 것을 의미합니다.

- 일반적으로 각 Consumer Group은 별도의 애플리케이션이고 Consumer는 AMQP 프로토콜을 사용하여 Event Hub의 이벤트를 읽는 Consumer Group 내의 프로세스입니다. 이상적인 것은 확장을 제공하기 위해 파티션 수만큼 Consumer를 가지는 것이지만 파티션보다 더 많은 Consumer를 가질 수 있으며, 이 경우 각 Consumer는 동일한 파티션에서 이벤트를 읽습니다. 이는 권장되는 것은 아니며 이 경우 중복 데이터를

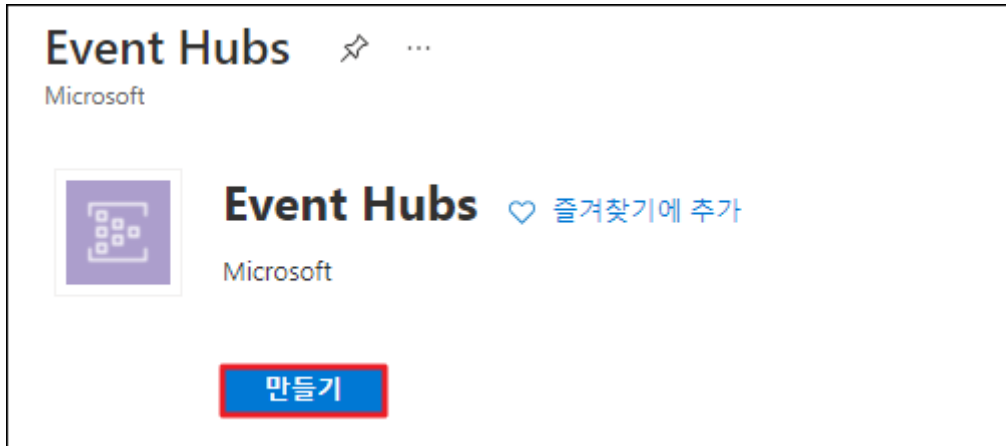
직접 처리해야 합니다.

- 오프셋(Offset)은 파티션 내의 이벤트 위치이며 파티션의 각 이벤트에는 오프셋이 있습니다. 오프셋은 이벤트의 byte 번호이며, 클라이언트 측의 커서로 생각할 수 있습니다. 오프셋을 사용하여 Event Consumer가 이벤트를 읽어야 할 이벤트 스트림의 위치를 지정할 수 있습니다. 오프셋은 여러 파티션에 저장할 수 있으므로 각 Consumer는 현재 데이터를 처리하는 위치를 알 수 있습니다.
- 오프셋을 저장하는 과정을 체크포인팅(Checkpointing)라고 하며 클라이언트 측에서 수행됩니다. 체크포인트 메커니즘을 만들면 Consumer가 중지되거나 프로세스가 중지되었을 때 가장 마지막 오프셋으로부터 시작할 수 있기 때문에 더 많은 안정성을 확보할 수 있습니다. 체크포인트 작업은 클라이언트 측에서 수행되기 때문에 SDK를 사용하는 경우 기본적으로 이 기능을 사용할 수 있으며 Azure Blob에 대한 연결 문자열을 제공하면 SDK가 자동으로 체크포인트를 관리합니다.
- Consumer Group은 데이터 및 애플리케이션에 대한 보기일 뿐이므로 여러 Consumer Group을 가질 수 있습니다. 각 Consumer Group은 자체 체크포인트 메커니즘을 통해 자체 오프셋을 저장하여 데이터에 대한 자체 보기를 갖게 됩니다. Event Hub는 데이터를 1일~7일까지 저장할 수 있으므로 각 Consumer는 해당 기간 내에 데이터를 처리해야 합니다.



TASK 01. Event Hub 만들기

1. Azure 포털에서 [리소스 만들기]를 클릭한 후 "Event Hubs"를 검색합니다. [Event Hubs] 블레이드에서 [만들기]를 클릭합니다.



2. [네임스페이스 만들기] 블레이드의 [기본 사항] 탭에서 아래와 같이 구성한 후 [다음]을 클릭합니다.
 - [프로젝트 세부 정보 - 리소스 그룹]: "새로 만들기"를 클릭한 후 "02_eventHubsRg"를 입력합니다.
 - [인스턴스 세부 정보 - 네임스페이스 이름]: 중복되지 않는 고유한 이름을 입력합니다.
 - [인스턴스 세부 정보 - 위치]: Korea Central
 - [인스턴스 세부 정보 - 가격 책정 계층]: "표준"을 선택합니다. "기본"은 1개의 Consumer Group, "표준"은 20개의 Consumer Group, "프리미엄"은 100개의 Consumer Group을 지원합니다. 또한 각 가격 계층마다 서로 다른 메시지 보존 기간과 캡처 기능을 제공합니다.
 - [인스턴스 세부 정보 - 처리량 단위]: "1"을 선택합니다. 처리량 단위는 Event Hubs의 성능 단위를 설정하여 실제로 처리할 수 있는 메시지 수를 의미합니다. 최대 40개를 지정할 수 있습니다.
 - [인스턴스 세부 정보 - 자동 팅창 사용]: 선택하지 않음

네임스페이스 만들기 ...
Event Hubs

기본 사항 **네트워킹** 태그 검토 + 만들기

프로젝트 세부 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 폴더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

구독 *

리소스 그룹 * [새로 만들기](#)

인스턴스 세부 정보

가격 계층, 단위 수(용량) 구성과 같은 이 네임스페이스에 대한 필수 설정을 입력합니다.

네임스페이스 이름 * ☒ .servicebus.windows.net

위치 * ☒ 선택한 지역은 가용 영역을 지원합니다. 네임스페이스에서는 가용 영역이 활성화됩니다. [자세히 알아보세요.](#)

가격 책정 계층 * ☒ 사용할 수 있는 계획과 해당 기능 찾아보기

처리량 단위 *

자동 팽창 사용 ☐

3. [네트워킹] 탭에서 "공용 액세스"를 선택하고 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.

네임스페이스 만들기 ...
Event Hubs

기본 사항 **네트워킹** 태그 검토 + 만들기

네트워크 연결

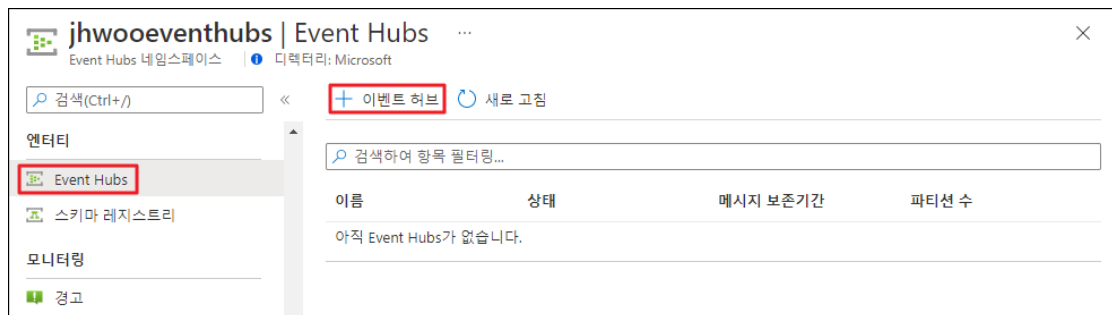
이 네임스페이스에 공용 IP 주소나 서비스 엔드포인트를 통해 공개적으로 또는 프라이빗 엔드포인트를 사용하여 비공개로 연결할 수 있습니다.

연결 방법 ☒ 공용 액세스 ☐ 프라이빗 액세스

4. 새로 만든 [Event Hubs 네임스페이스] 블레이드로 이동합니다. [설정 - 스케일링]으로 이동한 후 다음과 같은 내용을 검토합니다.
- 처리량 단위: Event Hubs 네임스페이스를 만들 때 구성할 수 있는 값이 표시됩니다.
 - 자동 팽창(Auto-Inflate): 다른 Azure 서비스의 자동 크기 조정과 유사한 기능입니다. 처리량 단위를 요구 사항에 맞게 자동으로 조정할 수 있는 옵션을 제공합니다.



5. 이제 Event Hubs 네임스페이스에 새 Event Hub를 만들어야 합니다. [Event Hubs 네임스페이스] 블레이드의 [엔터티 - Event Hubs]로 이동한 후 [이벤트 허브]를 클릭합니다.



6. [이벤트 허브 만들기] 블레이드에서 아래와 같이 구성한 후 [만들기]를 클릭합니다.
- 이름: my-demo
 - 파티션 수: "1"을 선택합니다. 파티션은 애플리케이션을 사용하는데 필요한 다운스트림 병렬 처리와 관련된 데이터 구성 메커니즘입니다.
 - 메시지 보존기간: "1"일을 선택합니다. Event Hub는 데이터 스트림이고 여러 애플리케이션이 데이터를 읽을 수 있기 때문에 메시지가 처리를 위해 Event Hub에서 보관될 일수를 지정합니다. "1"일을 선택하게 되면 24시간이 지난 데이터가 주기적으로 삭제됩니다. 메시지 보존 기간은 가격 계층에 따라 다릅니다.
 - 캡처: 해제

이벤트 허브 만들기 ...

Event Hubs

이름 * ⓘ
 ✓

파티션 수 ⓘ
 1

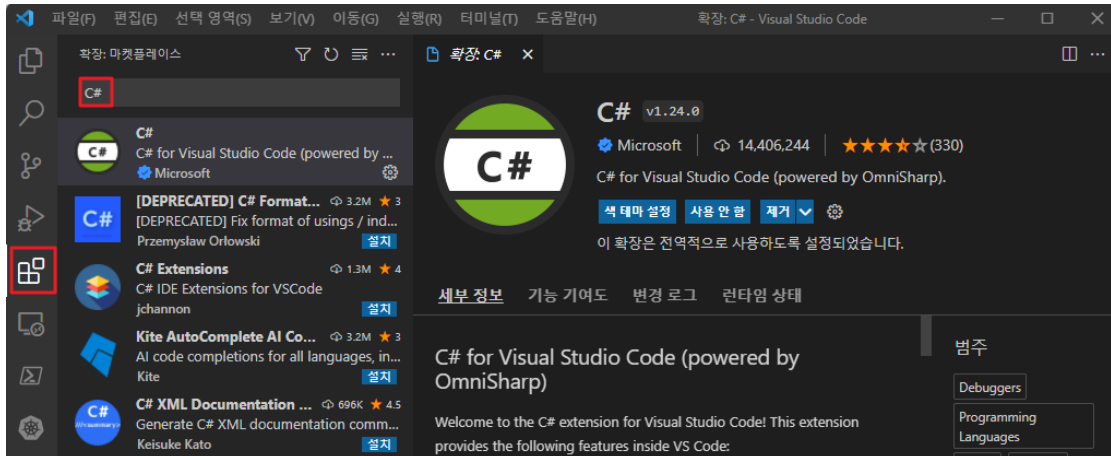
메시지 보존기간 ⓘ
 1

캡처 ⓘ

TASK 02. 데이터 출력 애플리케이션 만들기

이 작업에서는 Event Hub에 데이터를 출력하는 애플리케이션을 만듭니다.

1. Visual Studio Code를 열고 [확장] 메뉴를 클릭한 후 검색창에서 "C#"을 검색합니다. C# 확장에서 [설치]를 클릭하여 C# 확장을 설치합니다.



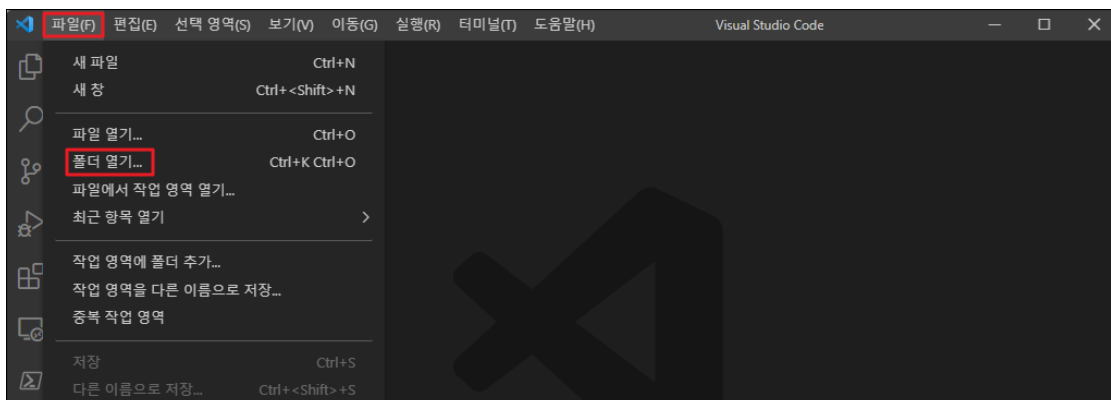
2. 브라우저를 열고 <https://dotnet.microsoft.com/en-us/download/dotnet/6.0> 사이트로 이동한 후 .NET SDK 6 최신 버전을 다운로드하고 설치합니다.

Build apps - SDK ①

SDK 6.0.101

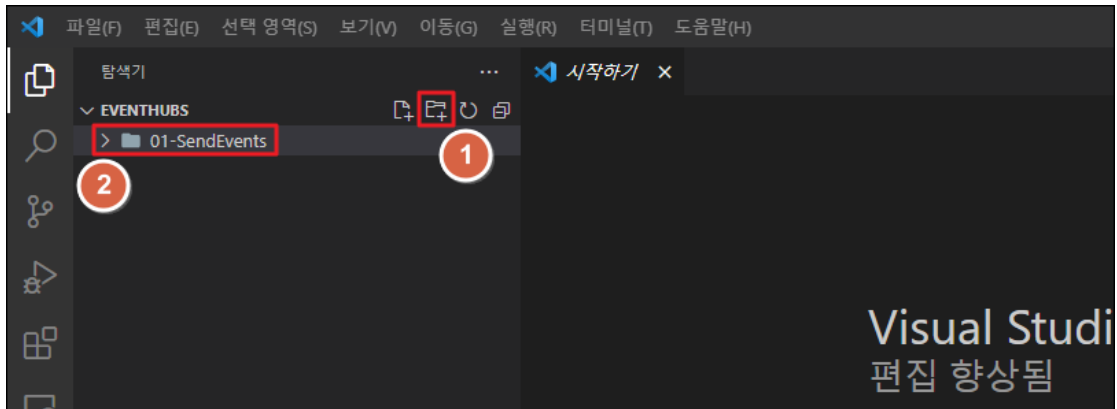
OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	Arm64 x64 x86	Arm64 x64 x86
All	dotnet-install scripts	

3. Visual Studio Code의 메뉴에서 [파일 - 폴더 열기]를 클릭합니다. 원하는 드라이브에 "EventHubs" 폴더를 만들고 이 폴더를 선택합니다.

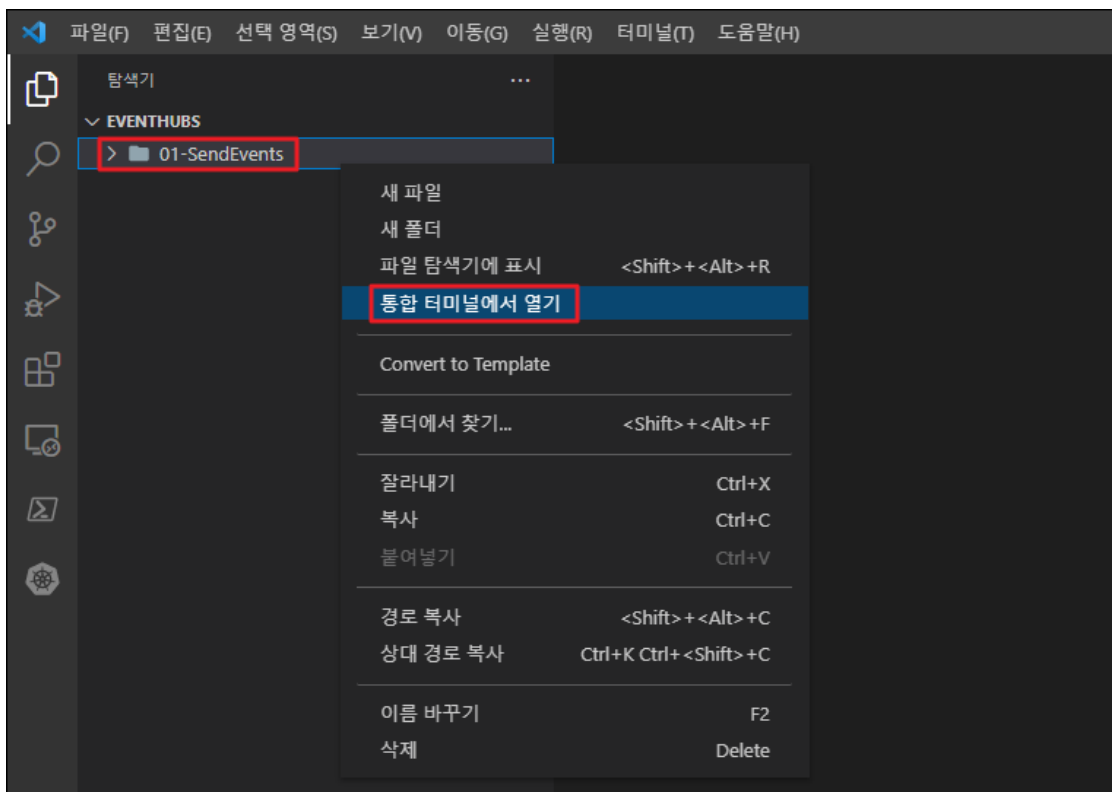


4. Visual Studio Code의 [탐색기]에서 [새 폴더] 아이콘을 클릭한 후 "01-SendEvents" 이름의 새 폴더를

만듭니다.



5. 새로 만든 폴더를 마우스 우 클릭한 후 [통합 터미널에서 열기]를 클릭합니다.



6. Visual Studio Code 터미널 창에서 프로젝트를 초기화하기 위해 다음 명령을 실행합니다.

```
# 프로젝트 초기화
dotnet new console
```

```

문제   출력   디버그 콘솔   터미널

PS D:\EventHubs\01-SendEvents> # 프로젝트 초기화
PS D:\EventHubs\01-SendEvents> dotnet new console
"콘솔 앱" 템플릿이 성공적으로 생성되었습니다.

생성 후 작업 처리 중...
D:\EventHubs\01-SendEvents\01-SendEvents.csproj에서 'dotnet restore' 실행 중 ...
  복원할 프로젝트를 확인하는 중...
  D:\EventHubs\01-SendEvents\01-SendEvents.csproj을(를) 82 ms 동안 복원했습니다.
  복원에 성공했습니다.

PS D:\EventHubs\01-SendEvents>

```

7. 새로 초기화한 콘솔 프로젝트에 패키지를 추가하기 위해 다음 명령을 실행합니다. 이벤트를 보내는 콘솔 애플리케이션이기 때문에 `EventHubs.Producer` 패키지도 추가합니다.

프로젝트에 패키지 추가

```
dotnet add package Azure.Messaging.EventHubs
dotnet add package Azure.Messaging.EventHubs.Producer
```

```

문제   출력   디버그 콘솔   터미널

PS D:\EventHubs\01-SendEvents> # 프로젝트에 패키지 추가
PS D:\EventHubs\01-SendEvents> dotnet add package Azure.Messaging.EventHubs
  복원할 프로젝트를 확인하는 중...
  Writing C:\Users\Jinhwan\AppData\Local\Temp\tmpE871.tmp
  info : 'D:\EventHubs\01-SendEvents\01-SendEvents.csproj' 프로젝트에 'Azure.Messaging.EventHubs' 패키지에 대한 PackageReference를 추가하는 중입니다.
  info :  CACHE https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs/index.json
  info : D:\EventHubs\01-SendEvents\01-SendEvents.csproj의 패키지를 복원하는 중...
  info : 'Azure.Messaging.EventHubs' 패키지는 'D:\EventHubs\01-SendEvents\01-SendEvents.csproj' 프로젝트에 지정된 모든 프레임워크와 호환됩니다.
  info : 'Azure.Messaging.EventHubs' 패키지 '5.6.2' 버전에 대한 PackageReference가 'D:\EventHubs\01-SendEvents\01-SendEvents.csproj' 파일에 추가되었습니다.
  info : 복원을 커밋하는 중...
  info : 자산 파일을 디스크에 쓰는 중입니다. 경로: D:\EventHubs\01-SendEvents\obj\project.assets.json
  log : D:\EventHubs\01-SendEvents\01-SendEvents.csproj을(를) 258 ms 동안 복원했습니다.
PS D:\EventHubs\01-SendEvents> dotnet add package Azure.Messaging.EventHubs.Producer
  복원할 프로젝트를 확인하는 중...
  Writing C:\Users\Jinhwan\AppData\Local\Temp\tmpFC85.tmp
  info : 'D:\EventHubs\01-SendEvents\01-SendEvents.csproj' 프로젝트에 'Azure.Messaging.EventHubs.Producer' 패키지에 대한 PackageReference를 추가하는 중입니다.
  info :  GET https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.producer/index.json
  info :  NotFound https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.producer/index.json 909밀리초
  info :  GET https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.producer/index.json
  info :  NotFound https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.producer/index.json 322밀리초
  error: 'Azure.Messaging.EventHubs.Producer' 패키지에 사용할 수 있는 버전이 없습니다.

```

8. 프로젝트의 종속성 및 도구를 복원하여 모든 패키지가 다운로드 되었는지 확인하기 위해 다음 명령을 실행합니다.

패키지 확인

```
dotnet restore
```

```

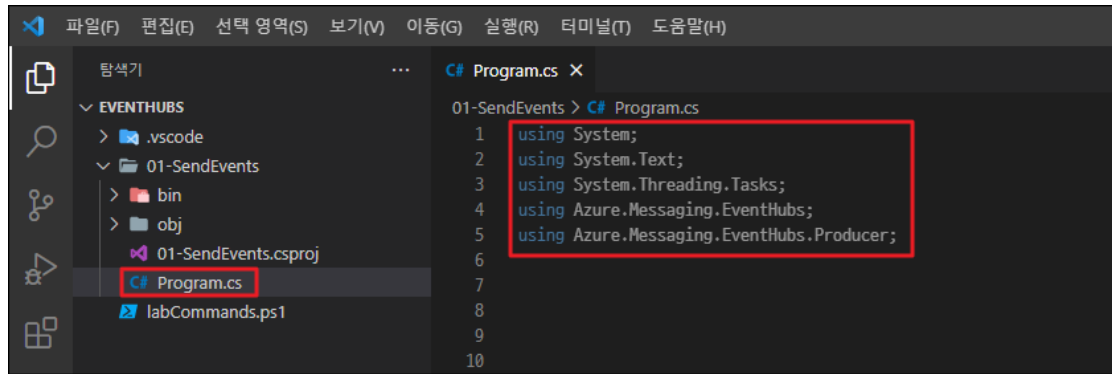
문제   출력   디버그 콘솔   터미널

PS D:\EventHubs\01-SendEvents> # 패키지 확인
PS D:\EventHubs\01-SendEvents> dotnet restore
  복원할 프로젝트를 확인하는 중...
  복원할 모든 프로젝트가 최신 상태입니다.
PS D:\EventHubs\01-SendEvents>

```

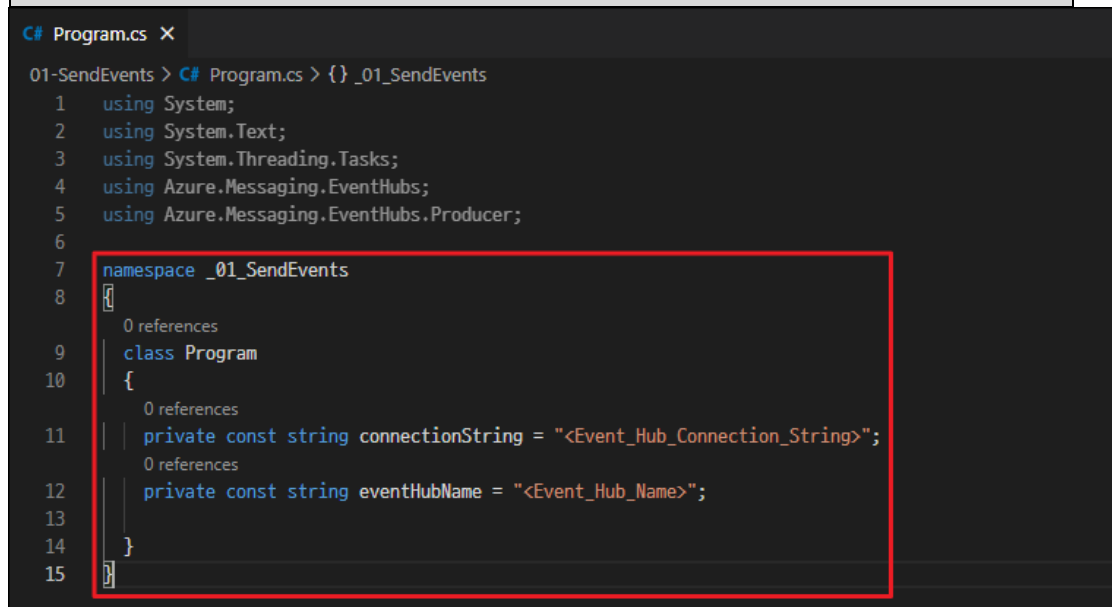
9. Visual Studio Code의 [탐색기]에서 "`01-SendEvents\Program.cs`" 파일을 열고 기본 입력 내용을 모두 삭제합니다. 다음과 같은 `using` 문을 추가합니다.

```
using System;
using System.Text;
using System.Threading.Tasks;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Producer;
```

10. Event Hub 연결 문자열과 Event Hub 이름에 대한 속성을 다음과 같이 정의합니다.

```
namespace _01_SendEvents
{
    class Program
    {
        private const string connectionString = "<Event_Hub_Connection_String>";
        private const string eventHubName = "<Event_Hub_Name>";
    }
}
```



11. 다음과 같은 코드를 추가합니다.

- Event Hub용 클라이언트를 위해 `EventHubProducerClient`를 만듭니다.
- 배치 데이터 패키지를 만들고 여러 이벤트를 단일 배치로 업로드한 후 이를 Event Hub에 단일 메시지로 전송합니다.
- `TryAdd` 메서드를 사용하여 이벤트 데이터 클래스를 전달하고 UTF-8로 인코딩합니다. 이를 통해 Event Hub에 3개의 이벤트를 전송합니다.
- 마지막으로 콘솔 화면에 결과를 출력하도록 설정합니다.

```
static async Task Main()
{
    // Event Hub 로 이벤트를 보내는데 사용할 수 있는 Producer Client 만들기
```

```

    await using (var producerClient = new EventHubProducerClient(connectionString,
eventHubName))
    {
        // 이벤트 배치 만들기
        using EventDataBatch eventBatch = await producerClient.CreateBatchAsync();

        // 배치에 이벤트 추가. 이벤트는 byte 와 메타데이터의 컬렉션으로 표현됨
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("First event")));
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Second event")));
        eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Third event")));

        // 이벤트의 배치를 Event Hub 로 전송하기 위해 Producer Client 사용
        await producerClient.SendAsync(eventBatch);
        Console.WriteLine("A batch of 3 events has been published.");
    }
}

```

```

11 | private const string connectionString = "<Event_Hub_Connection_String>";
    | 1 reference
12 | private const string eventHubName = "<Event_Hub_Name>";
    | 0 references
13 | static async Task Main()
    | {
14 |     {
    |     // Event Hub로 이벤트를 보내는데 사용할 수 있는 Producer Client 만들기
16 |     await using (var producerClient = new EventHubProducerClient(connectionString, eventHubName))
    |     {
17 |         {
    |         // 이벤트 배치 만들기
19 |         using EventDataBatch eventBatch = await producerClient.CreateBatchAsync();
    |
20 |         // 배치에 이벤트 추가. 이벤트는 byte와 메타데이터의 컬렉션으로 표현됨
22 |         eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("First event")));
23 |         eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Second event")));
24 |         eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes("Third event")));
    |
25 |         // 이벤트의 배치를 Event Hub로 전송하기 위해 Producer Client 사용
27 |         await producerClient.SendAsync(eventBatch);
28 |         Console.WriteLine("A batch of 3 events has been published.");
    |     }
29 |     }
    | }
30 | }
    | }
31 |
32 | }

```

12. 이제 애플리케이션이 준비되었으므로 Event Hub 연결 문자열과 Event Hub 이름을 코드에 입력해야 합니다. Azure 포털로 이동한 후 [Event Hubs 네임스페이스] 블레이드의 [엔터티 - Event Hubs]로 이동합니다. 앞서 만들었던 Event Hub 이름(my-demo)을 복사합니다. my-demo 인스턴스를 클릭합니다.

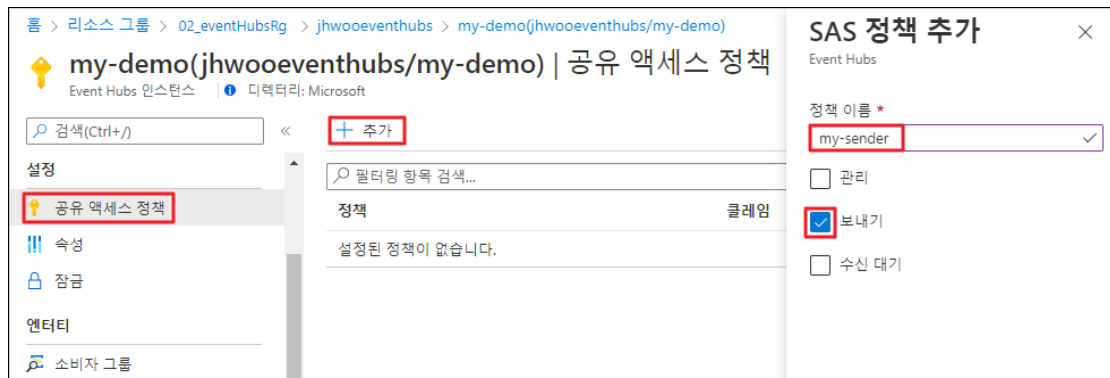
The screenshot shows the Azure Event Hubs portal for the namespace 'jhwooeventhubs'. The left sidebar has 'Event Hubs' highlighted. The main area shows a table with one instance, 'my-demo', which is 'Active' and has a '1' partition count. The 'my-demo' name in the table is highlighted with a red box.

이름	상태	메시지 보존기간	파티션 수
my-demo	Active	1 일	1

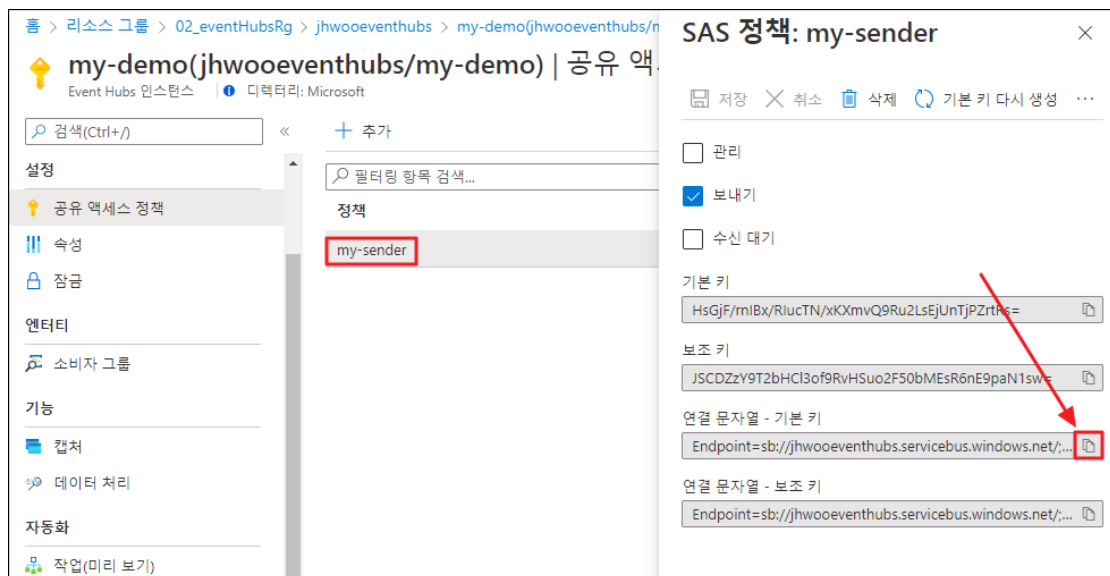
13. [my-demo Event Hubs 인스턴스] 블레이드의 [설정 - 공유 액세스 정책]으로 이동한 후 [추가]를

클릭합니다. [SAS 정책 추가]에서 아래와 같이 구성한 후 [만들기]를 클릭합니다.

- 정책 이름: my-sender
- "보내기" 정책만 선택합니다.



14. [설정 - 공유 액세스 정책]에서 위에서 만든 정책이 표시되는 것을 확인합니다. 이 정책을 클릭합니다. [SAS 정책]에서 "연결 문자열 - 기본 키"에 표시되는 내용을 클립보드로 복사합니다.



15. Visual Studio Code로 이동한 후 Event Hub 이름과 연결 문자열에 위에서 확인했던 값을 붙여 넣습니다.

- **connectionString**: SAS 정책에서 복사한 연결 문자열을 붙여 넣습니다. 이 연결 문자열은 Event Hubs 네임스페이스가 아니라 Event Hub 자체의 연결 문자열이기 때문에 연결 문자열의 마지막 **EntityPath**에 Event Hub 이름이 표시됩니다.
- **eventHubName**: Event Hub의 이름(**my-demo**)를 붙여 넣습니다.

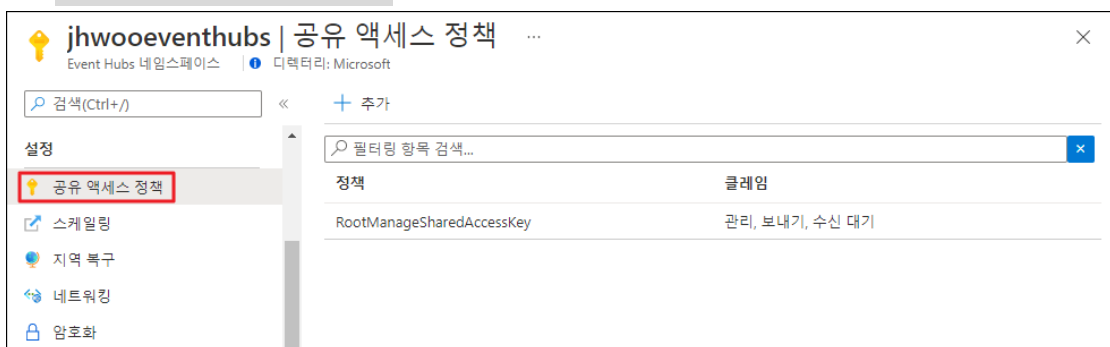
```

7 namespace _01_SendEvents
8 {
9     class Program
10    {
11        private const string connectionString = "Endpoint=sb://jhwooeventhubs.servicebus.windows.net;/SharedAccessKeyName=my-sender;";
12        private const string eventHubName = "my-demo";
13        static async Task Main()
14        {
15            // Event Hub로 이벤트를 보내는데 사용할 수 있는 Producer Client 만들기
16            await using (var producerClient = new EventHubProducerClient(connectionString, eventHubName))
17            {
18                // 이벤트 배치 만들기
19                using EventDataBatch eventBatch = await producerClient.CreateBatchAsync();
20            }
21        }
22    }
23 }

```

16. Azure 포털로 이동한 후 [Event Hubs 네임스페이스] 블레이드의 [설정 - 공유 액세스 정책]으로 이동합니다.

- 여기에서 Event Hubs 네임스페이스의 모든 Event Hub에 대한 공유 액세스 정책을 확인할 수 있습니다. 따라서 여러 Event Hub에 데이터를 업로드할 때 이 위치에서 공유 액세스 정책을 만들 수 있습니다.
- RootManageSharedAccessKey**는 전체 Event Hub에 대한 관리 키입니다.



17. Visual Studio Code로 전환한 후 다음 명령을 실행하여 애플리케이션을 빌드합니다.

```
# 애플리케이션 빌드
dotnet build
```

```

문제    출력    디버그 콘솔    터미널

PS D:\EventHubs\01-SendEvents> # 애플리케이션 빌드
PS D:\EventHubs\01-SendEvents> dotnet build
.NET용 Microsoft (R) Build Engine 버전 17.0.0+c9eb9dd64
Copyright (C) Microsoft Corporation. All rights reserved.

복원할 프로젝트를 확인하는 중...
복원할 모든 프로젝트가 최신 상태입니다.
01-SendEvents -> D:\EventHubs\01-SendEvents\bin\Debug\net6.0\01-SendEvents.dll

빌드했습니다.
경고 0개
오류 0개

경과 시간: 00:00:02.88
PS D:\EventHubs\01-SendEvents>

```

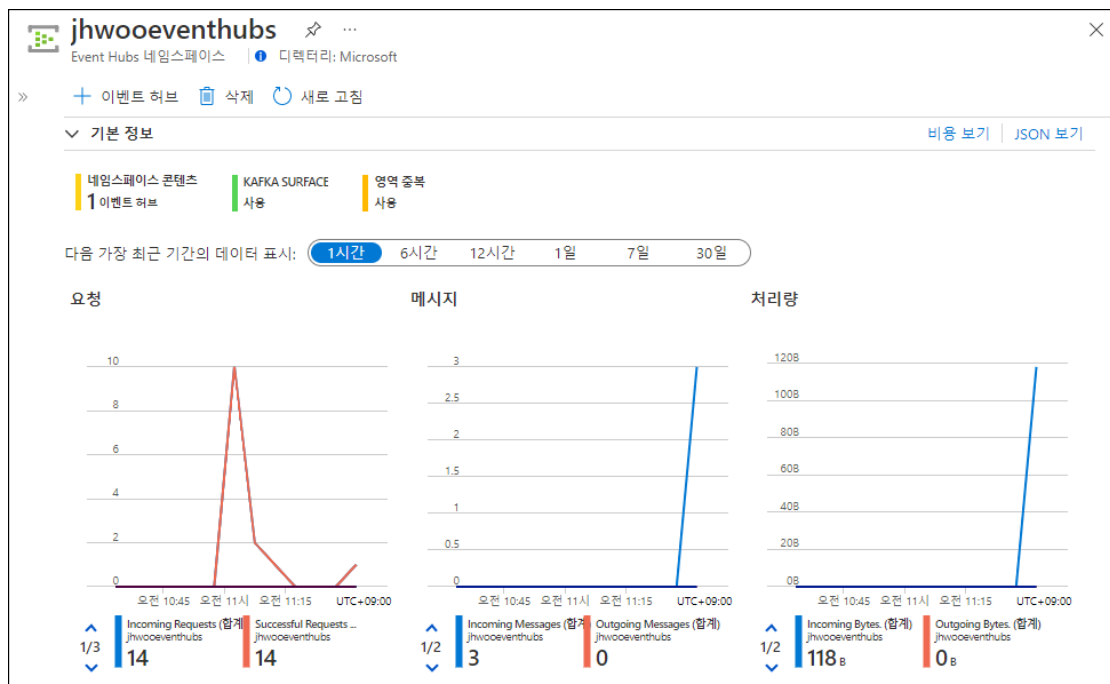
18. 다음 명령을 실행하여 애플리케이션을 실행합니다. 애플리케이션 실행 후 3개의 이벤트가 게시되었다는 메시지가 표시되는지 확인합니다. 즉 3개의 이벤트를 배치를 사용하여 Event Hub에 게시하였습니다.

```
# 애플리케이션 실행  
dotnet run
```

문제 출력 디버그 콘솔 터미널

```
PS D:\EventHubs\01-SendEvents> # 애플리케이션 실행  
PS D:\EventHubs\01-SendEvents> dotnet run  
A batch of 3 events has been published.  
PS D:\EventHubs\01-SendEvents> █
```

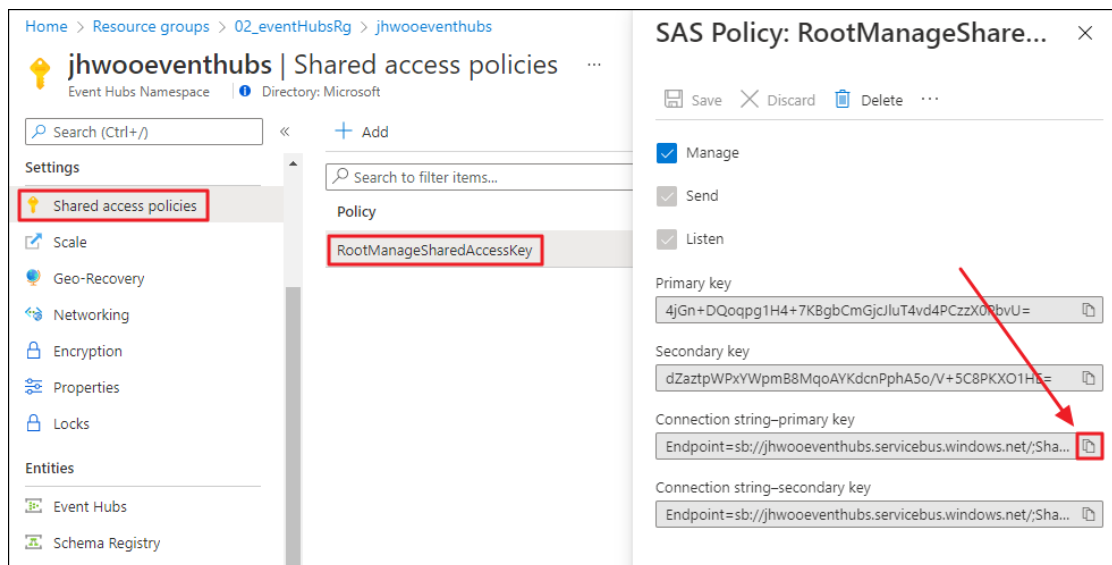
19. Event Hub에 대한 메트릭을 확인하기 위해 Azure 포털로 이동합니다. [Event Hubs 네임스페이스] 블레이드의 [개요]에서 네임스페이스 전체의 메트릭을 검토할 수 있습니다. 마찬가지로 특정 Event Hub에서도 메트릭 데이터를 확인할 수 있습니다.



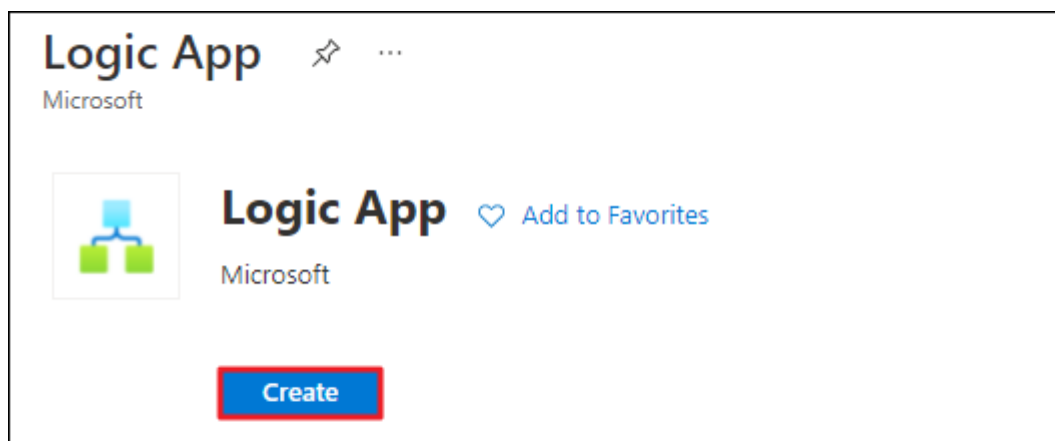
TASK 03. Logic Apps를 통해 Event Hub로 이벤트 전송

이 작업에서는 Logic Apps를 통해 Event Hub로 이벤트를 전송하는 방법에 대해 실습합니다. Logic Apps의 여러 작업을 검색할 때 한국어를 사용하기 어렵기 때문에 Azure 포털의 언어를 영어로 변경한 후 작업을 진행합니다.

1. 우선 Logic App에서 Event Hub에 연결할 때 사용할 연결 문자열을 확인해야 합니다. [Event Hubs Namespace] 블레이드의 [Settings - Shared access policies]로 이동한 후 **RootManageSharedAccessKey**를 클릭합니다. [SAS Policy]에서 **Connection string-primary key** 값을 복사합니다.



2. Azure 포털에서 [Create a resource]를 클릭한 후 "Logic App"을 검색하고 클릭합니다. [Logic App] 블레이드에서 [Create]를 클릭합니다.



3. [Create Logic App] 블레이드의 [Basics] 탭에서 아래와 같이 구성한 후 [Review + create]를 클릭합니다. [Review + create] 탭에서 [Create]를 클릭합니다.
 - [Project Details - Resource Group]: 02_eventHubsRg
 - [Instance Details - Type]: Consumption
 - [Instance Details - Logic App name]: logicApp-sendevents
 - [Instance Details - Region]: Korea Central

- [Instance Details - Enable log analytics]: No

Create Logic App

Basics Tags Review + create

Create a logic app, which lets you group workflows as a logical unit for easier management, deployment and sharing of resources. Workflows let you connect your business-critical apps and services with Azure Logic Apps, automating your workflows without writing a single line of code.

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 Internal

Resource Group * ⓘ 02_eventHubsRg
[Create new](#)

Instance Details

Type * ☒ Consumption ☐ Standard
[Looking for the classic consumption create experience? Click here](#)

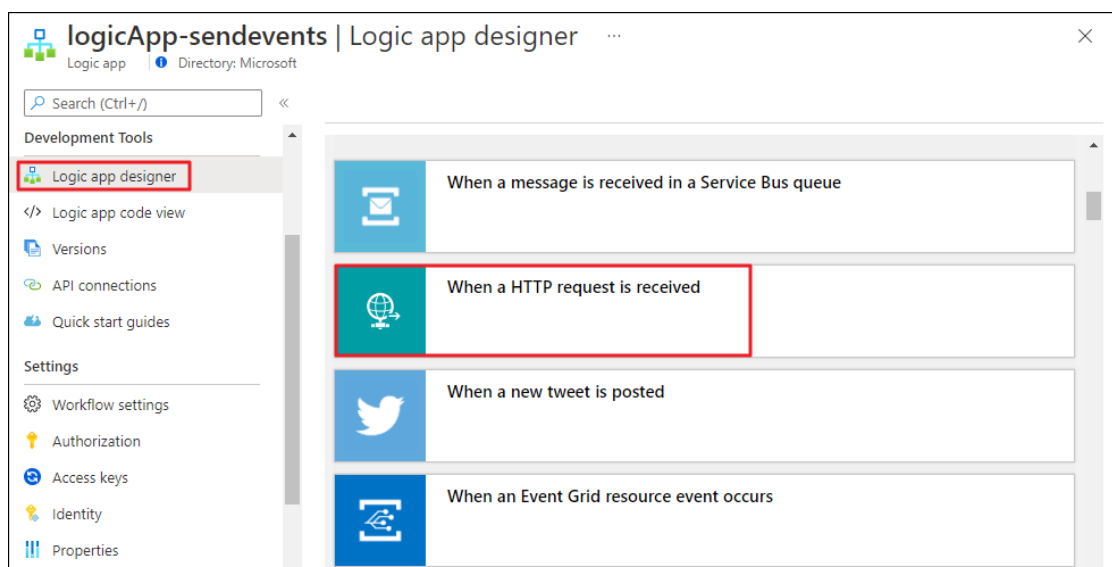
Logic App name * logicApp-sendevents ✓

Region * Korea Central ✓

Enable log analytics * ☐ Yes ☒ No

⚠ There are no log analytics workspace resources in the selected subscription. In order to enable log analytics, either create a new log analytics workspace resource or switch to a subscription which already has one.

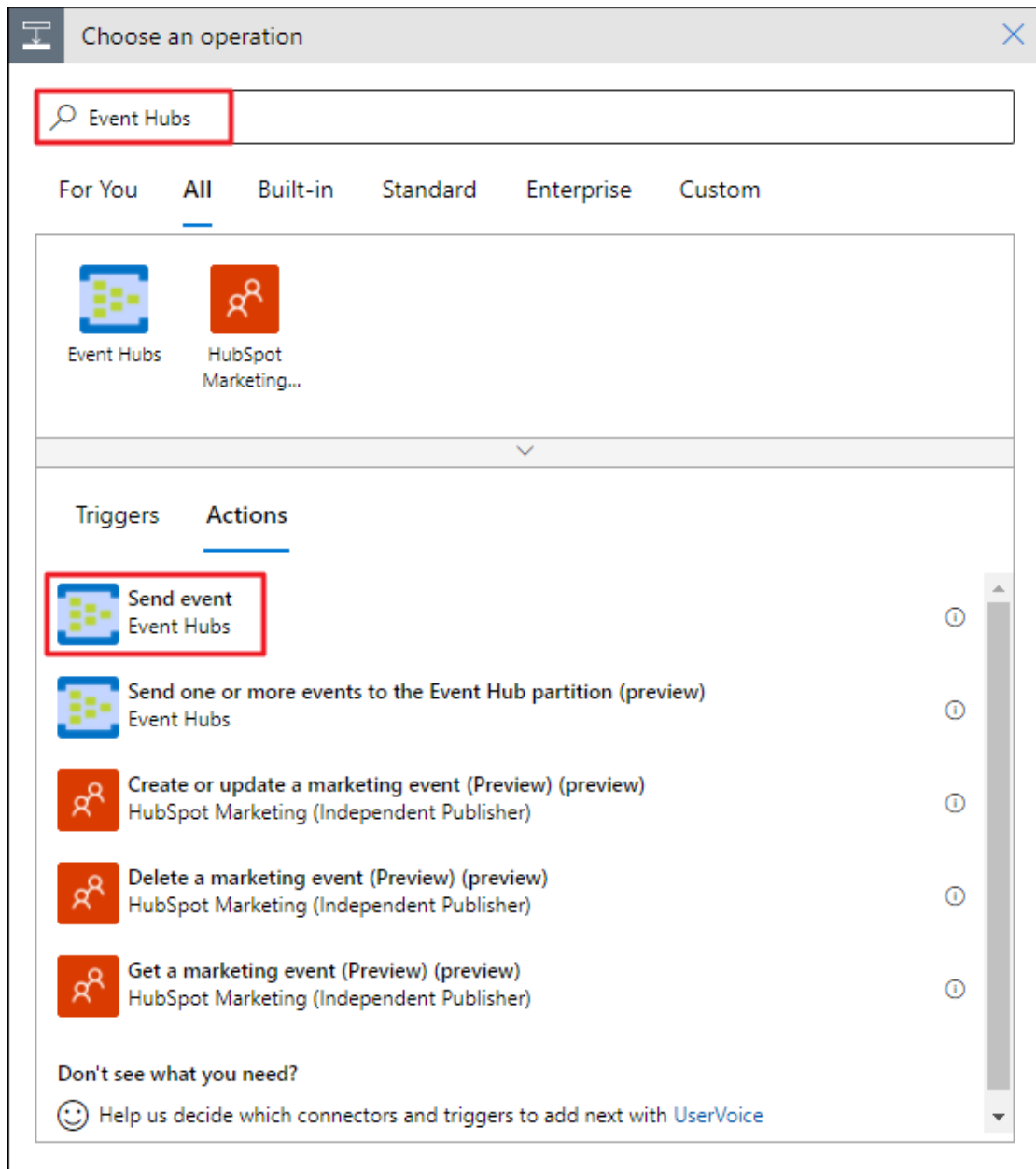
4. [Logic app] 블레이드의 [Development Tools - Logic app designer]로 이동한 후 [When a HTTP request is received] 타일을 클릭합니다.



5. Logic App 디자이너에서 [New step]을 클릭합니다.

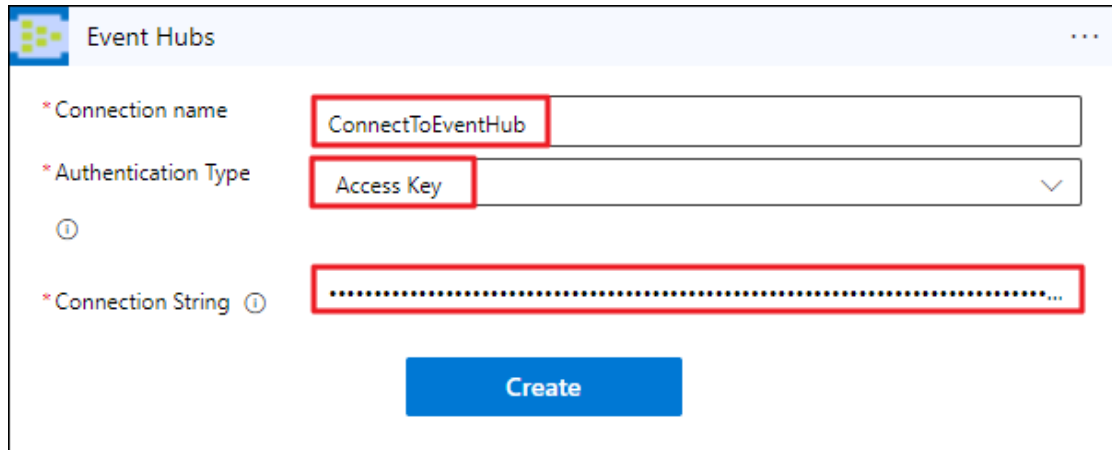
The screenshot shows the configuration for the 'When a HTTP request is received' trigger in Azure Logic Apps. The configuration includes a field for 'HTTP POST URL' with a placeholder 'URL will be generated after save', a 'Request Body JSON Schema' text area, a link 'Use sample payload to generate schema', and a dropdown menu 'Add new parameter'. At the bottom, there is a '+ New step' button highlighted with a red rectangle.

6. [Choose an operation] 타일에서 "Event Hubs"를 검색한 후 [Actions] 탭에서 "Send event"를 클릭합니다.



7. [Event Hubs] 타일에서 아래와 같이 구성한 후 [Create]를 클릭합니다.

- Connection name: ConnectToEventHub
- Authentication Type: Access Key
- Connection String: Event Hub에서 복사했던 연결 문자열을 붙여 넣습니다.



Event Hubs

* Connection name: ConnectToEventHub

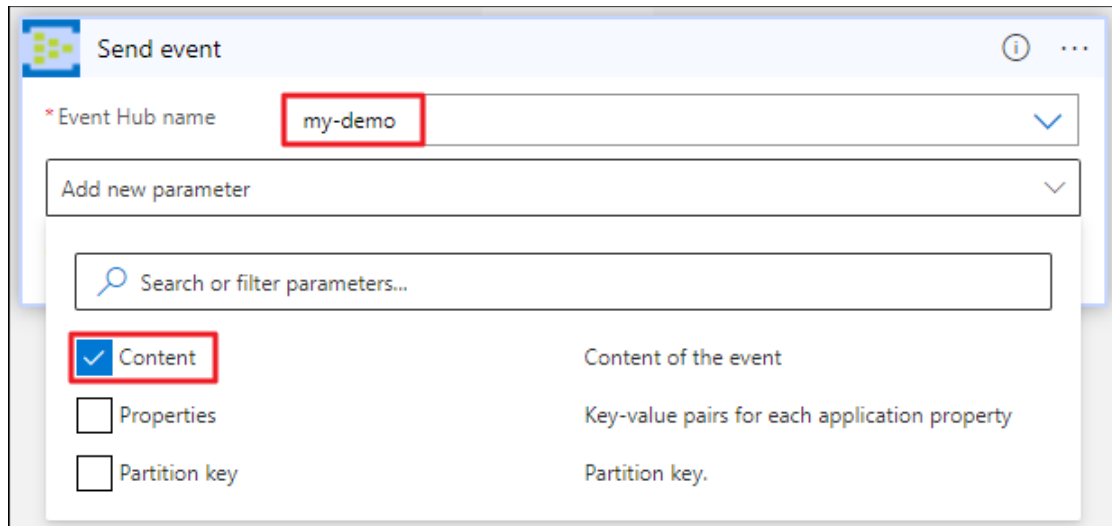
* Authentication Type: Access Key

* Connection String:

Create

8. [Send event] 타일에서 아래와 같이 구성합니다.

- Event Hub name: 드롭 다운 메뉴를 클릭한 후 "my-demo"를 선택합니다.
- Add new parameter: 클릭한 후 "Content"를 선택합니다. "Content"는 Event Hub에 보낼 수 있는 작은 메시지입니다.



Send event

* Event Hub name: my-demo

Add new parameter

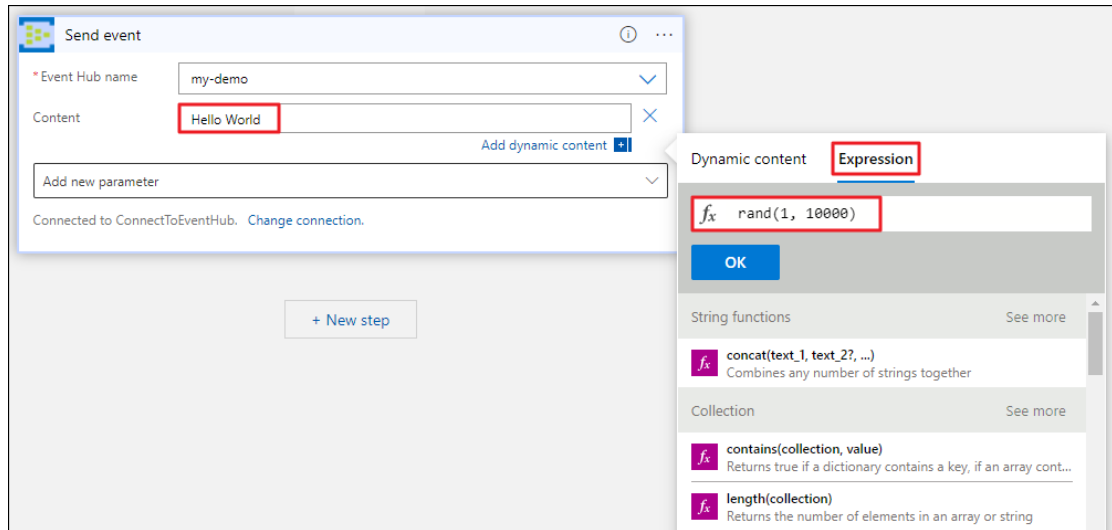
Search or filter parameters...

☒ Content: Content of the event

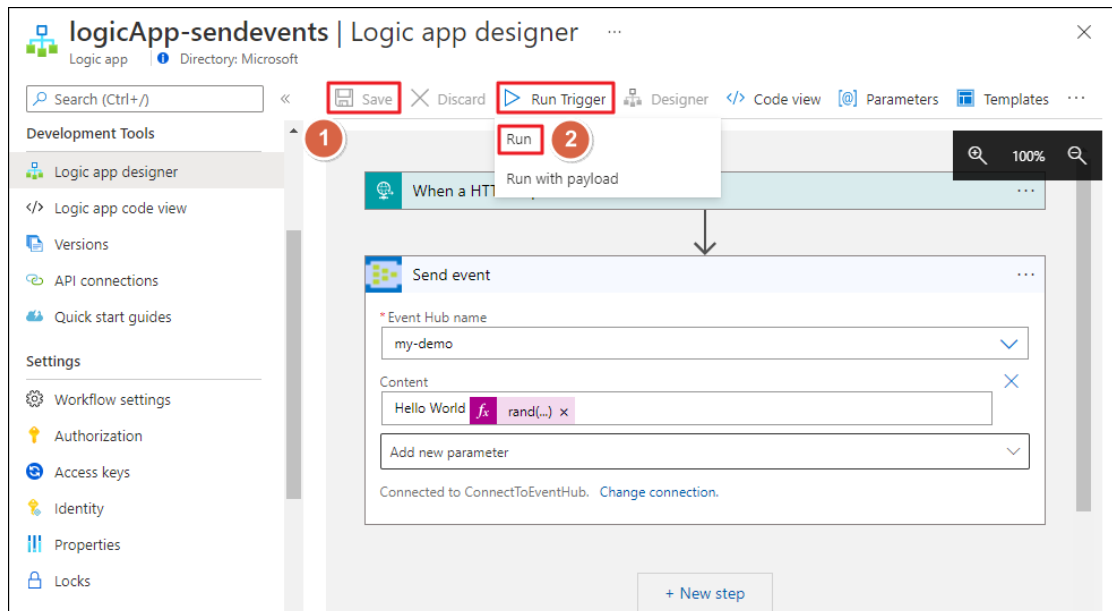
☐ Properties: Key-value pairs for each application property

☐ Partition key: Partition key.

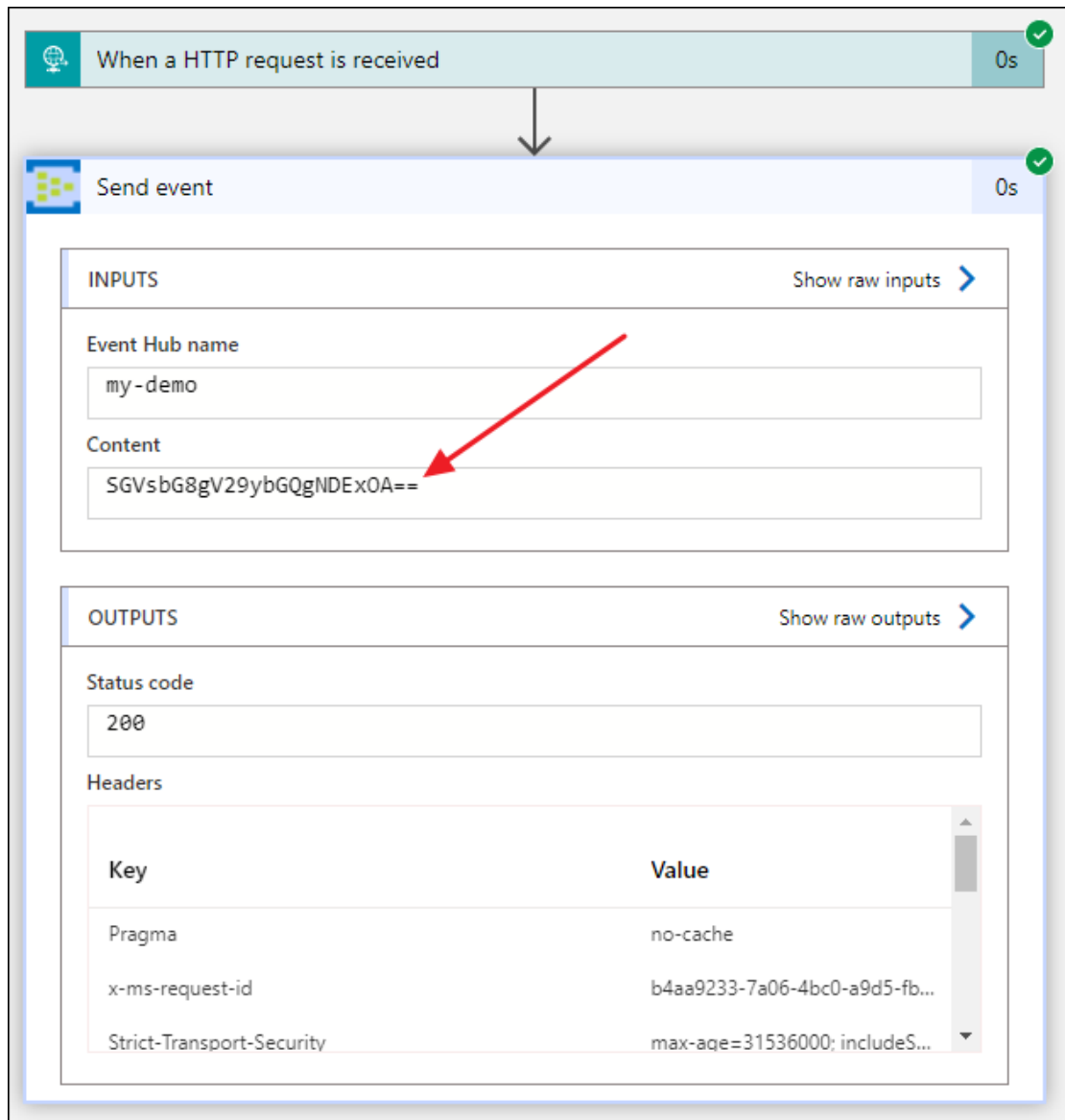
9. [Send event] 타일의 "Content"에 "Hello World"를 입력하고 동적 콘텐츠의 [Expression] 탭에서 `rand(1, 10000)`을 입력한 후 [OK]를 클릭합니다.



10. [Send event] 타일이 아래와 같이 구성된 것을 확인한 후 메뉴에서 [Save]를 클릭하고 [Run Trigger - Run]을 클릭합니다.



11. 다음과 같이 트리거 실행 결과를 확인할 수 있습니다. [Send event]를 확장하면 입력했던 콘텐츠가 자동으로 인코딩되어 표시되는 것을 확인할 수 있습니다.



When a HTTP request is received 0s ✓

Send event 0s ✓

INPUTS Show raw inputs >

Event Hub name
my-demo

Content
SGVsbG8gV29ybGQgNDExOA==

OUTPUTS Show raw outputs >

Status code
200

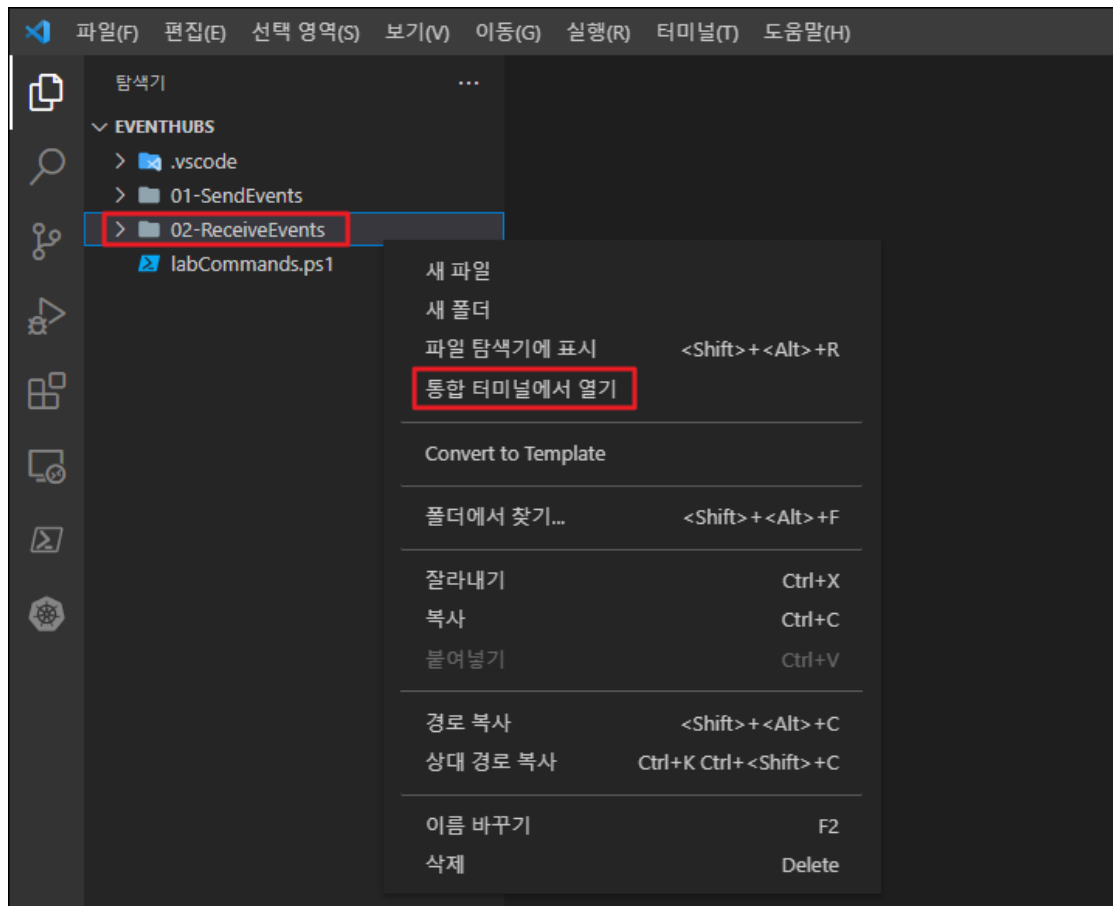
Headers

Key	Value
Pragma	no-cache
x-ms-request-id	b4aa9233-7a06-4bc0-a9d5-fb...
Strict-Transport-Security	max-age=31536000; includeS...

TASK 04. Event Hub 수신 애플리케이션 만들기

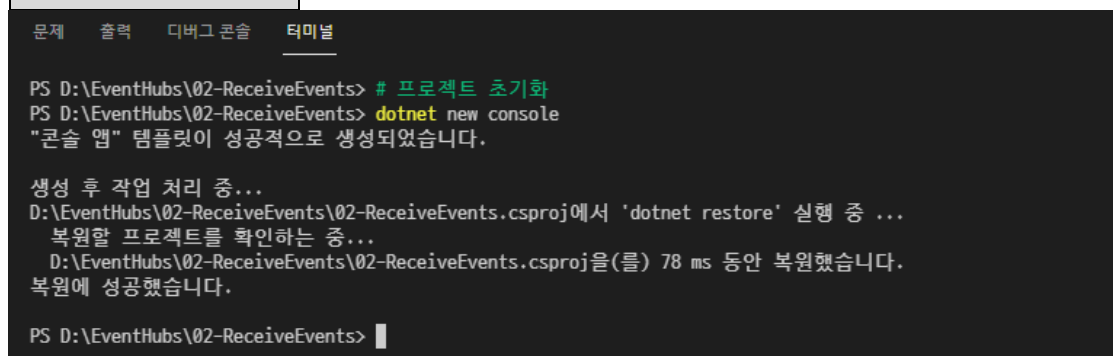
이 작업에서는 수신 이벤트 애플리케이션을 만듭니다.

1. Visual Studio Code의 [탐색기]에서 "02-ReceiveEvent" 이름의 폴더를 만듭니다. 새로 만든 폴더를 마우스 우 클릭하고 [통합 터미널에서 열기]를 클릭합니다.



2. 새 콘솔 애플리케이션을 만들기 위해 터미널 창에서 다음 명령을 실행합니다.

```
# 프로젝트 초기화
dotnet new console
```



3. 데이터 수신 프로젝트를 실행하는데 필요한 패키지를 추가하기 위해 다음 명령을 실행합니다.

```
# 프로젝트에 패키지 추가
dotnet add package Azure.Messaging.EventHubs
dotnet add package Azure.Messaging.EventHubs.Consumer
dotnet add package Azure.Messaging.EventHubs.Processor
```

```

문제   출력   디버그 콘솔   터미널

PS D:\EventHubs\02-ReceiveEvents> # 프로젝트에 패키지 추가
PS D:\EventHubs\02-ReceiveEvents> dotnet add package Azure.Messaging.EventHubs
복원할 프로젝트를 확인하는 중...
Writing C:\Users\Jinhwan\AppData\Local\Temp\tmp5F11.tmp
info : 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 'Azure.Messaging.EventHubs' 패키지에 대한 PackageReference를 추가하는 중입니다.
info : CACHE https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs/index.json
info : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj의 패키지를 복원하는 중...
info : 'Azure.Messaging.EventHubs' 패키지는 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 지정된 모든 프레임워크와 호환됩니다.
info : 'Azure.Messaging.EventHubs' 패키지 '5.6.2' 버전에 대한 PackageReference가 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 파일에 추가되었습니다.
info : 복원을 커밋하는 중...
info : 자산 파일을 디스크에 쓰는 중입니다. 경로: D:\EventHubs\02-ReceiveEvents\obj\project.assets.json
log : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj을(를) 240 ms 동안 복원했습니다.

PS D:\EventHubs\02-ReceiveEvents> dotnet add package Azure.Messaging.EventHubs.Consumer
복원할 프로젝트를 확인하는 중...
Writing C:\Users\Jinhwan\AppData\Local\Temp\tmp96FA.tmp
info : 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 'Azure.Messaging.EventHubs.Consumer' 패키지에 대한 PackageReference를 추가하는 중입니다.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.consumer/index.json
info : NotFound https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.consumer/index.json 945밀리초
info : GET https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.consumer/index.json
info : NotFound https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.consumer/index.json 313밀리초
error: 'Azure.Messaging.EventHubs.Consumer' 패키지에 사용할 수 있는 버전이 없습니다.

PS D:\EventHubs\02-ReceiveEvents> dotnet add package Azure.Messaging.EventHubs.Processor
복원할 프로젝트를 확인하는 중...
Writing C:\Users\Jinhwan\AppData\Local\Temp\tmpDC21.tmp
info : 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 'Azure.Messaging.EventHubs.Processor' 패키지에 대한 PackageReference를 추가하는 중입니다.
info : CACHE https://api.nuget.org/v3/registration5-gz-semver2/azure.messaging.eventhubs.processor/index.json
info : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj의 패키지를 복원하는 중...
info : 'Azure.Messaging.EventHubs.Processor' 패키지는 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 지정된 모든 프레임워크와 호환됩니다.
info : 'Azure.Messaging.EventHubs.Processor' 패키지 '5.6.2' 버전에 대한 PackageReference가 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 파일에 추가되었습니다.
info : 복원을 커밋하는 중...
info : 자산 파일을 디스크에 쓰는 중입니다. 경로: D:\EventHubs\02-ReceiveEvents\obj\project.assets.json
log : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj을(를) 198 ms 동안 복원했습니다.
PS D:\EventHubs\02-ReceiveEvents>

```

4. 모든 패키지가 정상적으로 추가된 것을 확인하기 위해 다음 명령을 실행합니다.

```
# 패키지 확인
dotnet restore
```

```

문제   출력   디버그 콘솔   터미널

PS D:\EventHubs\02-ReceiveEvents> # 패키지 확인
PS D:\EventHubs\02-ReceiveEvents> dotnet restore
복원할 프로젝트를 확인하는 중...
복원할 모든 프로젝트가 최신 상태입니다.
PS D:\EventHubs\02-ReceiveEvents>

```

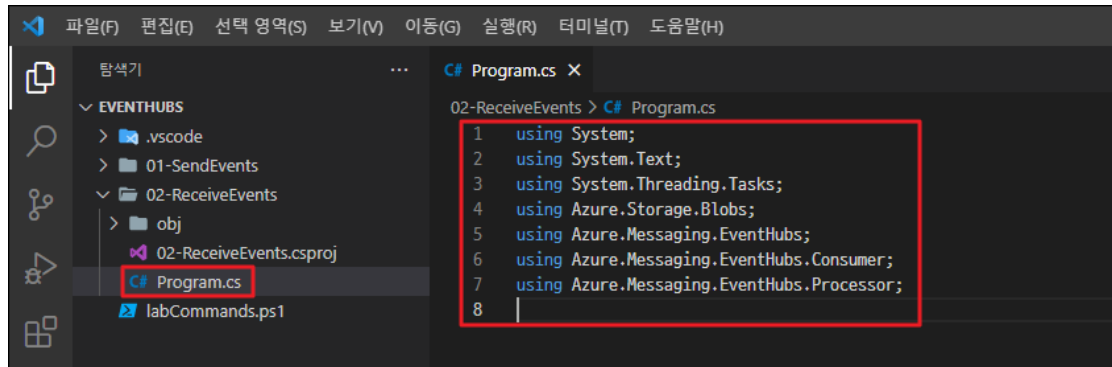
5. Visual Studio Code의 [탐색기]에서 "02-ReceiveEvents\Program.cs" 파일을 열고 코드창에 다음과 같은 `using` 문을 추가합니다.

- Consumer Group은 데이터를 처리할 때마다 체크포인트를 저장해야 하기 때문에 이를 위해 체크포인트 프로세스를 추가합니다.
- 체크포인트를 저장하기 위해 Azure Blob Storage에 해당 체크포인트를 저장하는 SDK 기능을 사용합니다.

```

using System;
using System.Text;
using System.Threading.Tasks;
using Azure.Storage.Blobs;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Consumer;
using Azure.Messaging.EventHubs.Processor;

```



6. 위에서 체크포인트 저장을 위해 Azure Blob Storage를 추가하였기 때문에 Blob Storage 용 패키지를 추가해야 합니다. 터미널 창에서 다음 명령을 실행하여 Blob Storage 용 패키지를 추가합니다.

```
# Azure Blob Storage 용 패키지 추가
dotnet add package Azure.Storage.Blobs
```

```
문제 출력 디버그 콘솔 터미널
PS D:\EventHubs\02-ReceiveEvents> # Azure Blob Storage 용 패키지 추가
PS D:\EventHubs\02-ReceiveEvents> dotnet add package Azure.Storage.Blobs
복원할 프로젝트를 확인하는 중...
Writing C:\Users\Jinhwan\AppData\Local\Temp\tmpB0C9.tmp
info : 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 'Azure.Storage.Blobs' 패키지에 대한 PackageReference를 추가하는 중입니다.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/azure.storage.blobs/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/azure.storage.blobs/index.json 1007밀리초
info : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj의 패키지를 복원하는 중...
info : 'Azure.Storage.Blobs' 패키지는 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 프로젝트에 지정된 모든 프레임워크와 호환됩니다.
info : 'Azure.Storage.Blobs' 패키지 '12.10.0' 버전에 대한 PackageReference가 'D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj' 파일에 추가되었습니다.
info : 복원을 커밋하는 중...
info : 자산 파일을 디스크에 쓰는 중입니다. 경로: D:\EventHubs\02-ReceiveEvents\obj\project.assets.json
log : D:\EventHubs\02-ReceiveEvents\02-ReceiveEvents.csproj을(를) 195 ms 동안 복원했습니다.
PS D:\EventHubs\02-ReceiveEvents>
```

7. Event Hub에서 데이터를 수신하기 위해 4개의 속성이 필요합니다. 4개의 속성은 Event Hub 연결 문자열, Event Hub 이름, 스토리지 연결 문자열, 체크포인트를 저장할 스토리지 blob 컨테이너 이름입니다. 코드창에 이 4개의 속성을 정의합니다.

```
namespace _02_ReceiveEvents
{
    class Program
    {
        private const string connectionString = "<Event_Hub_Connection_String>";
        private const string eventHubName = "<Event_Hub_Name>";
        private const string blobStorageConnectionString = "<Blob_Connection_String>";
        private const string blobContainerName = "<Container_Name>";
    }
}
```

```
C# Program.cs X
02-ReceiveEvents > C# Program.cs > {} _02_ReceiveEvents
1  using System;
2  using System.Text;
3  using System.Threading.Tasks;
4  using Azure.Storage.Blobs;
5  using Azure.Messaging.EventHubs;
6  using Azure.Messaging.EventHubs.Consumer;
7  using Azure.Messaging.EventHubs.Processor;
8
9  namespace _02_ReceiveEvents
10 {
11     0 references
12     class Program
13     {
14         0 references
15         private const string connectionString = "<Event_Hub_Connection_String>";
16         0 references
17         private const string eventHubName = "<Event_Hub_Name>";
18         0 references
19         private const string blobStorageConnectionString = "<Blob_Connection_String>";
20         0 references
21         private const string blobContainerName = "<Container_Name>";
22     }
23 }
```

8. 다음과 같은 코드를 작성합니다.

- ① Consumer Group을 만들고 기본 Consumer Group 이름(\$Default)을 사용합니다. 모든 Event Hub에는 기본 Consumer Group이 있으며 원하는 경우 더 많은 Consumer Group을 만들 수 있습니다.
- ② blob 컨테이너 클라이언트를 만들고 이 클라이언트는 이벤트 프로세서 클라이언트에 매개 변수로 전달되어 ③ 이 클라이언트를 사용하여 blob 스토리지에 체크포인트 정보를 저장합니다.
- ④ 이벤트를 처리를 위해 두 개의 이벤트 핸들러를 추가합니다. 이벤트 핸들러는 프로세싱 중 오류를 확인하기 위한 것입니다.
- ⑤ `await processor.StartProcessingAsync()`를 실행하고 프로세스를 10초 동안 실행한 후 모든 메시지를 수신하고 처리를 종료합니다.

```
static async Task Main()
{
    string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;

    // Event Processor 가 사용할 blob 컨테이너 클라이언트 만들기
    BlobContainerClient storageClient = new
    BlobContainerClient(blobStorageConnectionString, blobContainerName);

    // Event Hub 에서 이벤트를 처리할 Event Processor 클라이언트 만들기
    EventProcessorClient processor = new EventProcessorClient(storageClient,
    consumerGroup, connectionString, eventHubName);

    // 이벤트 처리와 예외 핸들링을 위한 핸들러 등록
    processor.ProcessEventAsync += ProcessEventHandler;
    processor.ProcessErrorAsync += ProcessErrorHandler;
}
```



```
// 프로세싱 시작
await processor.StartProcessingAsync();

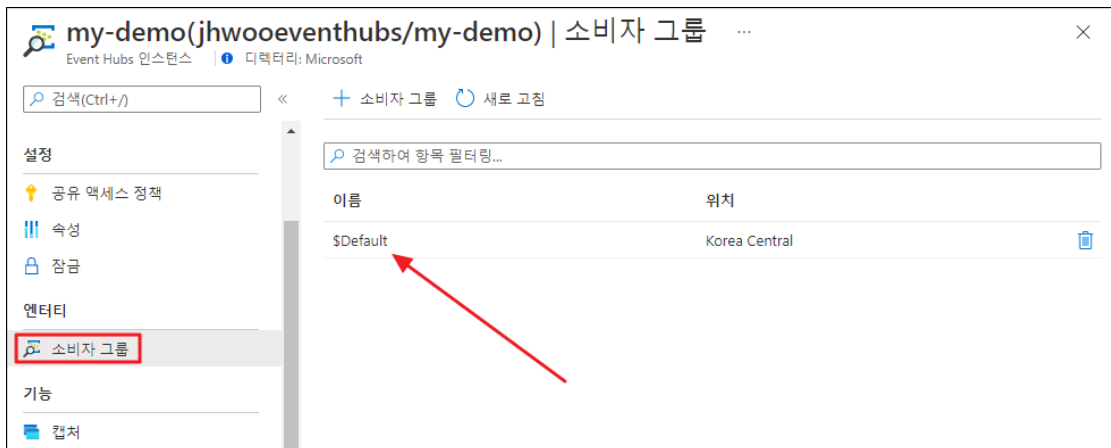
// 처리할 이벤트를 위해 10 초 동안 대기
await Task.Delay(TimeSpan.FromSeconds(10));

// 프로세싱 중지
await processor.StopProcessingAsync();
}
```

Program.cs

```
17 static async Task Main()
18 {
19     string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;
20
21     // Event Processor가 사용할 blob 컨테이너 클라이언트 만들기
22     BlobContainerClient storageClient = new BlobContainerClient(blobStorageConnectionString, blobContainerName);
23
24     // Event Hub에서 이벤트를 처리할 Event Processor 클라이언트 만들기
25     EventProcessorClient processor = new EventProcessorClient(storageClient, consumerGroup, connectionString, eventHubName);
26
27     // 이벤트 처리와 에러 핸들링을 위한 핸들러 등록
28     processor.ProcessEventAsync += ProcessEventHandler;
29     processor.ProcessErrorAsync += ProcessErrorHandler;
30
31     // 프로세싱 시작
32     await processor.StartProcessingAsync();
33
34     // 처리할 이벤트를 위해 10초 동안 대기
35     await Task.Delay(TimeSpan.FromSeconds(10));
36
37     // 프로세싱 중지
38     await processor.StopProcessingAsync();
39 }
```

9. Azure 포털에서 [Event Hubs 인스턴스] 블레이드의 [엔터티 - 소비자 그룹]으로 이동하면 아래와 같이 기본 Consumer Group이 있는 것을 확인할 수 있습니다.



10. 다음과 같은 코드를 추가합니다.

- ① `ProcessEventHandler`는 Event Hub에서 이벤트를 가져와 "Received event" 메시지와 이벤트 내용을 제공하고 이벤트 데이터를 디코딩합니다.
- ② `UpdateCheckpointAsync` 호출은 처리가 완료된 현재 오프셋으로 blob 스토리지를 업데이트합니다. 이 실습에서는 처리할 모든 단일 메시지에 대해 체크포인트를 업데이트합니다. 프로덕션 환경에서는 일반적으로 시간 간격을 기반으로 수행됩니다.
- ③ `ProcessErrorHandler`는 오류가 있는 경우 콘솔창에 오류 정보를 표시합니다.

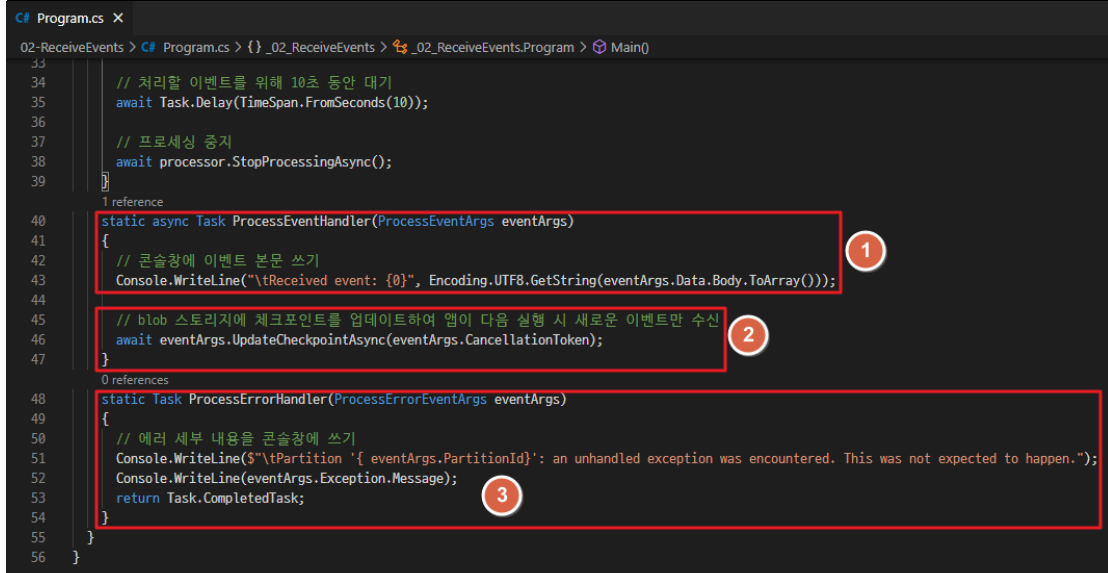
```

static async Task ProcessEventHandler(ProcessEventArgs eventArgs)
{
    // 콘솔창에 이벤트 본문 쓰기
    Console.WriteLine("\tReceived event: {0}",
        Encoding.UTF8.GetString(eventArgs.Data.Body.ToArray()));

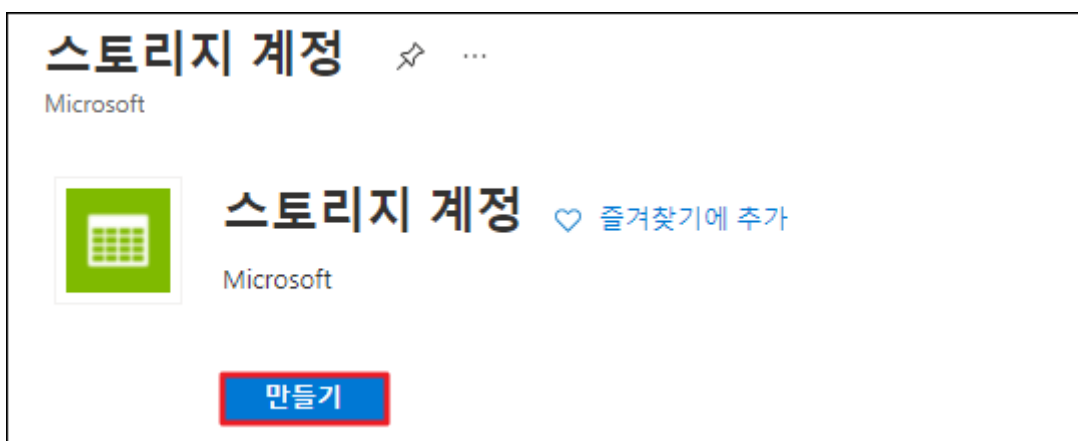
    // blob 스토리지에 체크포인트를 업데이트하여 앱이 다음 실행 시 새로운 이벤트만 수신
    await eventArgs.UpdateCheckpointAsync(eventArgs.CancellationToken);
}

static Task ProcessErrorHandler(ProcessErrorEventArgs eventArgs)
{
    // 에러 세부 내용을 콘솔창에 쓰기
    Console.WriteLine($"Partition '{eventArgs.PartitionId}': an unhandled exception
        was encountered. This was not expected to happen.");
    Console.WriteLine(eventArgs.Exception.Message);
    return Task.CompletedTask;
}

```



11. 이제 체크포인트에서 사용할 Azure 스토리지 계정을 만들어야 합니다. Azure 포털에서 [리소스 만들기]를 클릭한 후 "스토리지 계정"을 검색하고 클릭합니다. [스토리지 계정] 블레이드에서 [만들기]를 클릭합니다.



12. [저장소 계정 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성하고 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.

- [프로젝트 정보 - 리소스 그룹]: 02_eventHubsRg
- [인스턴스 정보 - 스토리지 계정 이름]: 중복되지 않는 고유한 이름을 입력합니다.
- [인스턴스 정보 - 지역]: (Asia Pacific) Korea Central
- [인스턴스 정보 - 성능]: 표준
- [인스턴스 정보 - 중복]: LRS(로컬 중복 스토리지)

저장소 계정 만들기

기본 고급 네트워킹 데이터 보호 암호화 태그 검토 + 만들기

Azure Storage는 가용성, 보안, 내구성, 확장성 및 중복성이 뛰어난 클라우드 스토리지를 제공하는 Microsoft 관리 서비스입니다. Azure Storage는 Azure Blob(개체), Azure Data Lake Storage Gen2, Azure Files, Azure 큐 및 Azure 테이블을 포함합니다. 스토리지 계정의 비용은 사용량 및 아래에서 선택한 옵션에 따라 다릅니다. [Azure Storage 계정에 대한 자세한 정보](#)

프로젝트 정보

새 스토리지 계정을 만들 구독을 선택합니다. 다른 리소스와 함께 스토리지 계정을 구성하고 관리할 새 리소스 그룹 또는 기존 리소스 그룹을 선택합니다.

구독 * Azure Internal

리소스 그룹 * 02_eventHubsRg
[새로 만들기](#)

인스턴스 정보

레거시 스토리지 계정 유형을 만들어야 하는 경우 다음을 클릭하세요. [여기](#).

스토리지 계정 이름 * jhwoochekpoint

지역 * (Asia Pacific) Korea Central

성능 * ☒ 표준: 대부분 시나리오에 권장됨(범용 v2 계정)
☐ 프리미엄: 짧은 대기 시간이 필요한 경우에 권장됩니다.

중복 * LRS(로컬 중복 스토리지)

13. 새로 만든 스토리지 계정 블레이드로 이동합니다. [데이터 스토리지 - 컨테이너]로 이동한 후 [컨테이너]를 클릭합니다. [새 컨테이너]에서 이름에 "capture"를 입력한 후 [만들기]를 클릭합니다.

홈 > 리소스 그룹 > 02_eventHubsRg > jhwoochekpoint

jhwoochekpoint | 컨테이너

스토리지 계정 | 디렉터리: Microsoft

검색(Ctrl+/)

데이터 스토리지

- 컨테이너
- 파일 공유
- 큐
- 테이블
- 보안 + 네트워킹
- 네트워킹

[+ 컨테이너](#) [액세스 수준 변경](#) [컨테이너 복원](#)

접두사로 컨테이너 검색

☒ 삭제된 컨테이너 표시

이름	마지막으로 수정한 ...	공용
\$logs	2022. 2. 2. 오후 8:56:...	프...

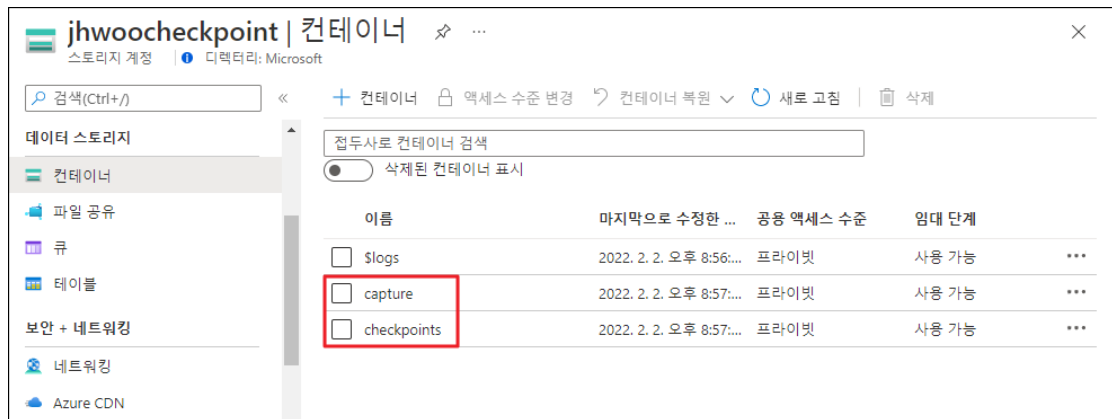
새 컨테이너

이름 * capture

공용 액세스 수준 * 프라이빗(익명 액세스 없음)

고급

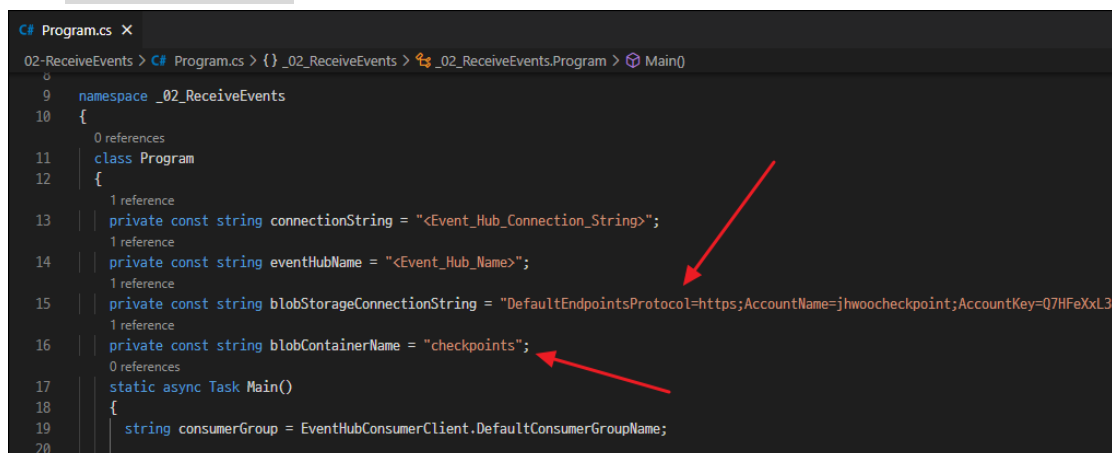
14. 동일한 방법으로 "checkpoints" 이름의 컨테이너도 추가합니다.



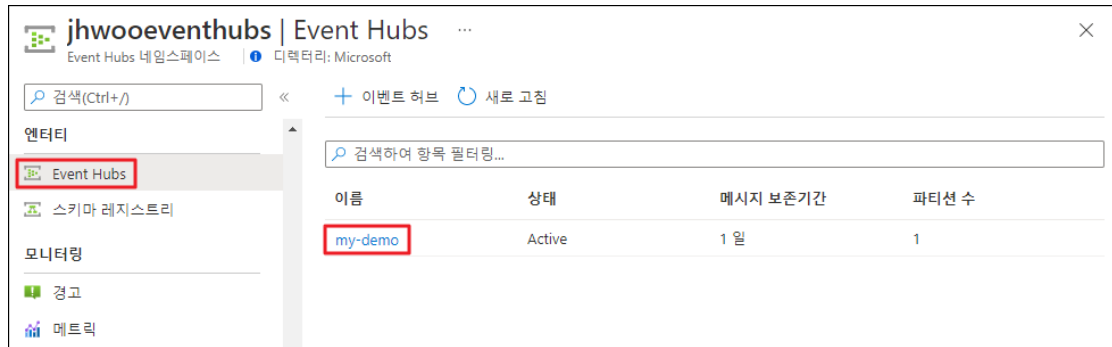
15. [스토리지 계정] 블레이드의 [보안 + 네트워킹 - 액세스 키]로 이동한 후 메뉴에서 [키 표시]를 클릭합니다. 표시되는 **key1**의 연결 문자열을 클립보드에 복사합니다.



16. Visual Studio Code로 이동한 후 `blobStorageConnectionString`에 위에서 복사한 연결 문자열을 붙여 넣고 `blobContainerName`에 "checkpoints"를 입력합니다.

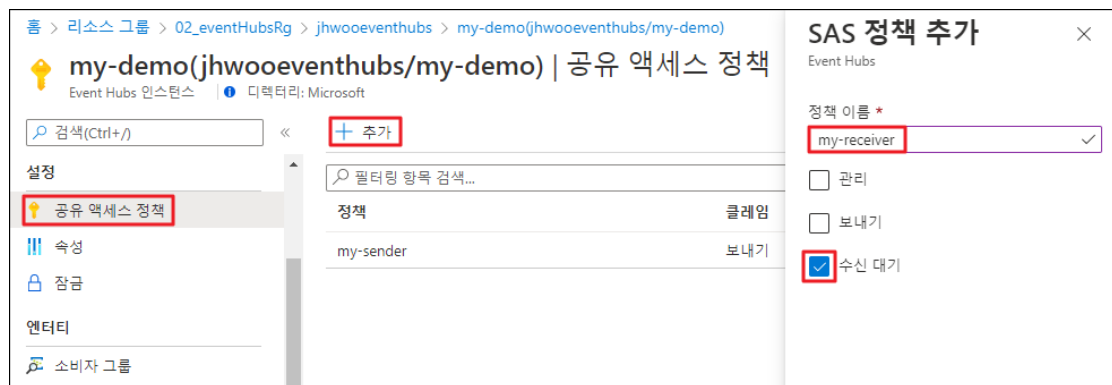


17. 이벤트 수신기에 사용할 정책을 만들어야 합니다. [Event Hubs 네임스페이스] 블레이드의 [엔터티 - Event Hubs]로 이동한 후 **my-demo** Event Hub를 클릭합니다.

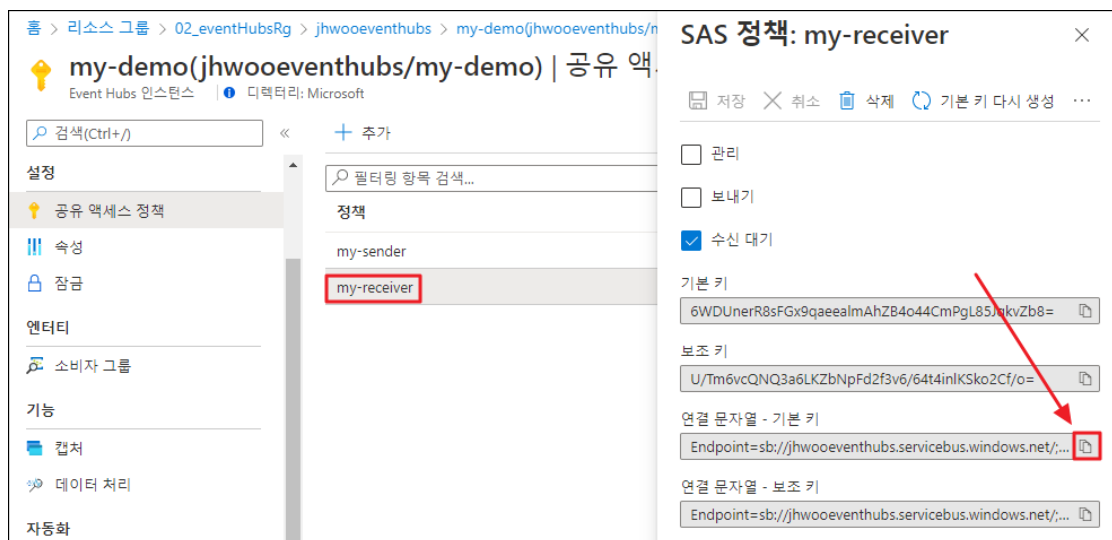


18. [my-demo Event Hubs 인스턴스] 블레이드의 [설정 - 공유 액세스 정책]으로 이동한 후 [추가]를 클릭합니다. [SAS 정책 추가]에서 아래와 같이 구성한 후 [만들기]를 클릭합니다.

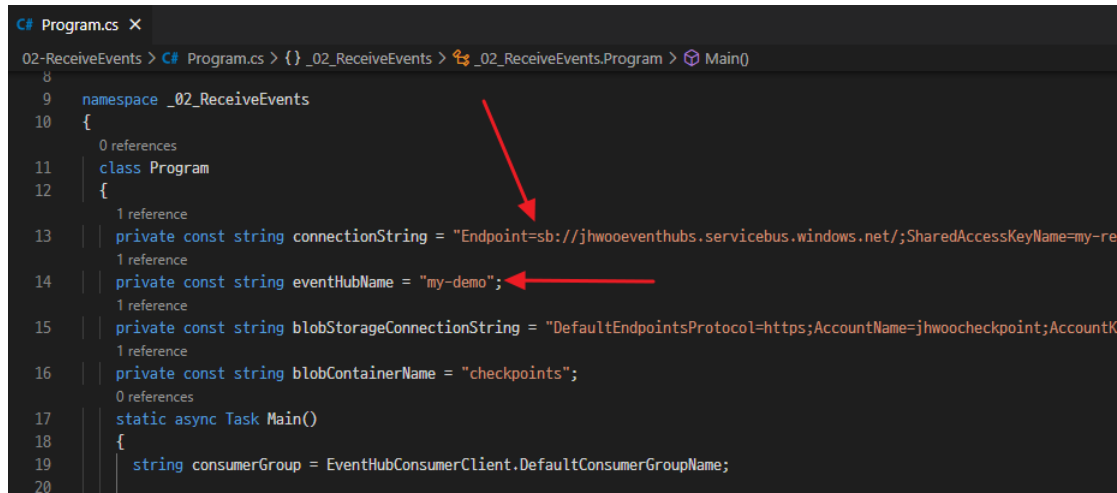
- 정책 이름: my-receiver
- "수신 대기"를 선택합니다.



19. [설정 - 공유 액세스 정책]에서 새로 만든 my-receiver 정책을 클릭합니다. [SAS 정책]에서 표시되는 "연결 문자열 - 기본 키" 값을 클립보드로 복사합니다.



20. Visual Studio Code로 이동한 후 connectionString에 위에서 복사한 연결 문자열을 붙여 넣고 eventHubName에 "my-demo"를 입력합니다.



```

C# Program.cs X
02-ReceiveEvents > C# Program.cs > {} _02_ReceiveEvents > _02_ReceiveEvents.Program > Main()
8
9 namespace _02_ReceiveEvents
10 {
11     0 references
12     class Program
13     {
14         1 reference
15         private const string connectionString = "Endpoint=sb://jhwooeventhubs.servicebus.windows.net/;SharedAccessKeyName=my-re
16         1 reference
17         private const string eventHubName = "my-demo";
18         1 reference
19         private const string blobStorageConnectionString = "DefaultEndpointsProtocol=https;AccountName=jhwoocheckpoint;AccountK
20         1 reference
21         private const string blobContainerName = "checkpoints";
22         0 references
23         static async Task Main()
24         {
25             string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;
26         }
27     }

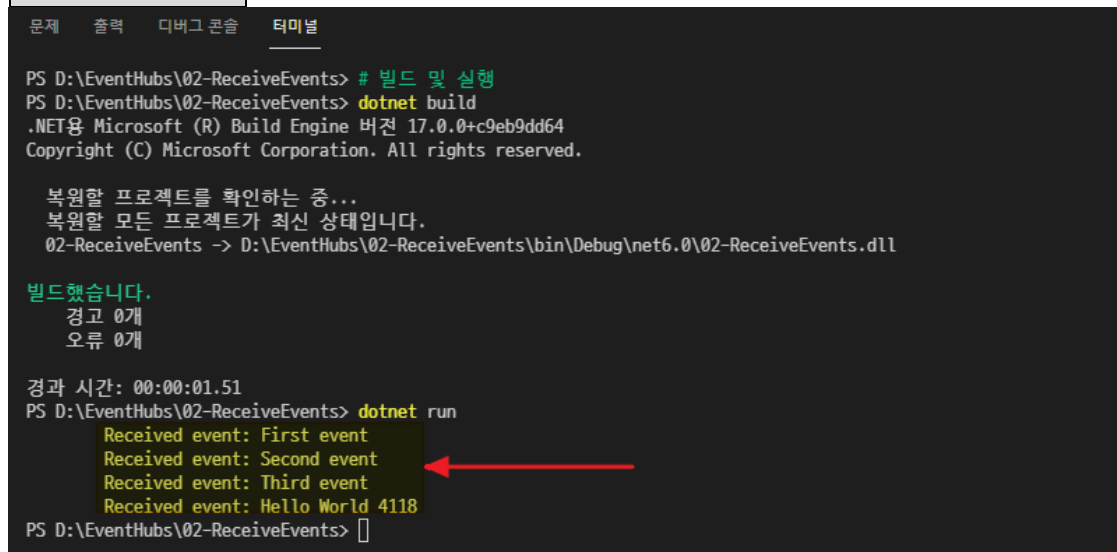
```

21. Visual Studio Code의 터미널에서 다음 명령을 실행하여 애플리케이션을 빌드하고 실행합니다.

- 실행 결과에서 애플리케이션에서 전송한 3개의 이벤트와 Logic App에서 보낸 1개의 이벤트가 출력되는 것을 확인할 수 있습니다.

빌드 및 실행

```
dotnet build
dotnet run
```



```

문제 출력 디버그 콘솔 터미널
PS D:\EventHubs\02-ReceiveEvents> # 빌드 및 실행
PS D:\EventHubs\02-ReceiveEvents> dotnet build
.NET용 Microsoft (R) Build Engine 버전 17.0.0+c9eb9dd64
Copyright (C) Microsoft Corporation. All rights reserved.

복원할 프로젝트를 확인하는 중...
복원할 모든 프로젝트가 최신 상태입니다.
02-ReceiveEvents -> D:\EventHubs\02-ReceiveEvents\bin\Debug\net6.0\02-ReceiveEvents.dll

빌드했습니다.
경고 0개
오류 0개

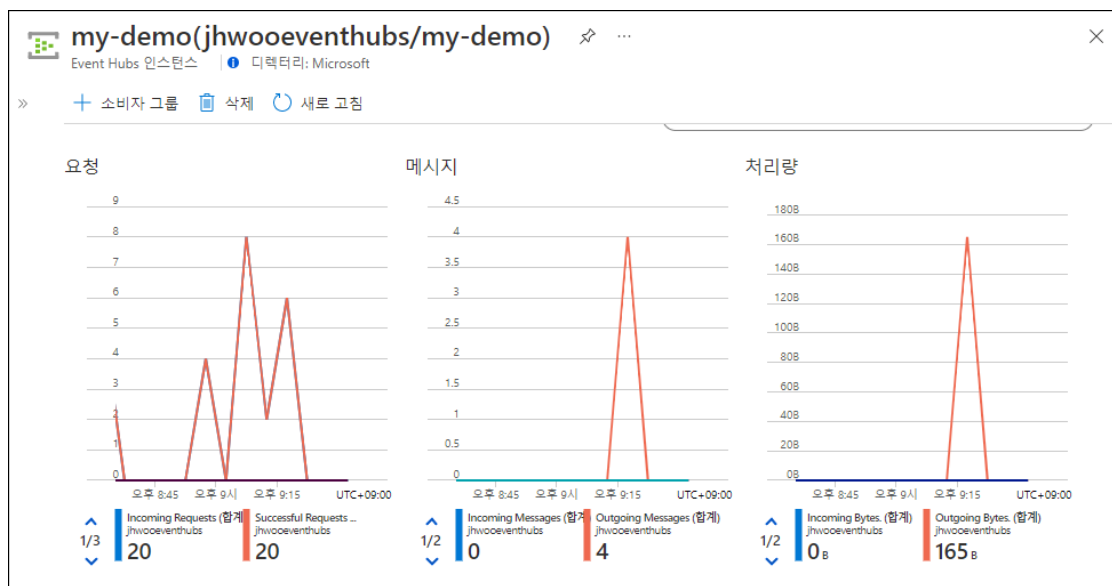
경과 시간: 00:00:01.51
PS D:\EventHubs\02-ReceiveEvents> dotnet run
Received event: First event
Received event: Second event
Received event: Third event
Received event: Hello World 4118
PS D:\EventHubs\02-ReceiveEvents>

```

22. Azure 포털에서 체크포인트 저장을 위해 만들었던 스토리지 계정으로 이동하면 다음과 같이 Event Hub 이름으로 체크포인트가 생성된 것을 확인할 수 있습니다.



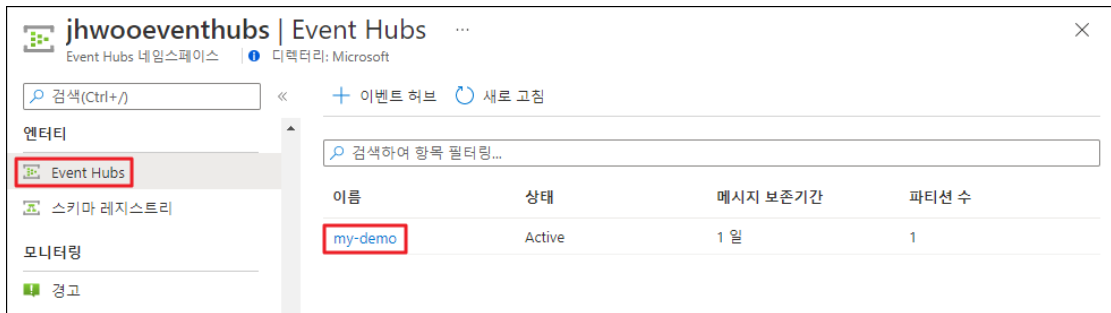
23. Azure 포털의 Event Hub 엔터티로 이동하면 아래와 같이 수신기에 의해 생성된 메트릭 데이터를 확인할 수 있습니다.



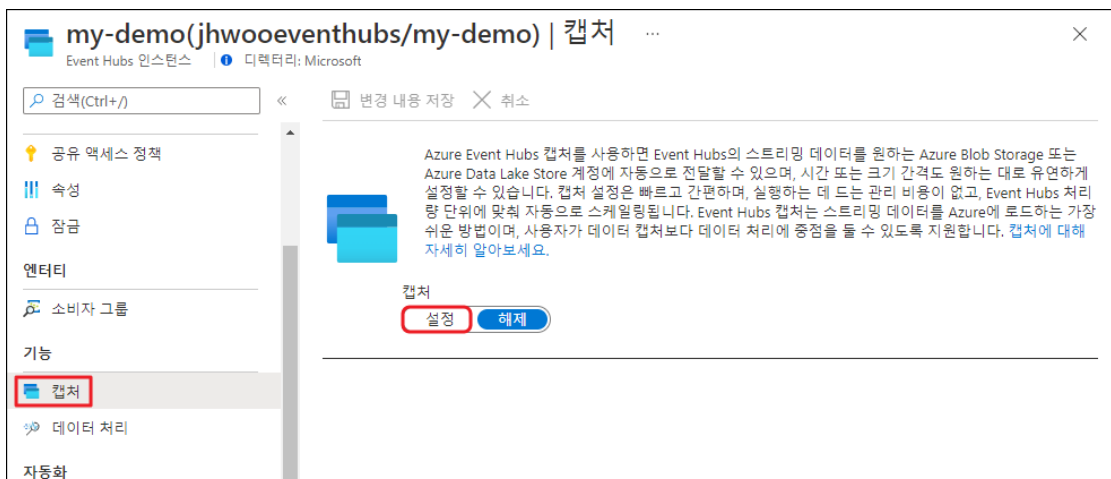
TASK 05. 이벤트 캡처

Event Hub를 만들 때 지정한 메시지 보존 기간 동안에 이벤트를 처리할 수 없거나 이벤트를 장기간 보존하려는 경우가 있습니다. 이 경우 이벤트 캡처 기능을 사용할 수 있습니다. 이벤트 캡처 기능은 Event Hub를 만들 때 사용하도록 설정할 수도 있습니다.

1. [Event Hubs 네임스페이스] 블레이드의 [엔터티 - Event Hubs]로 이동한 후 **my-demo** 인스턴스를 클릭합니다.



2. [**my-demo** Event Hubs 인스턴스] 블레이드의 [기능 - 캡처]로 이동한 후 캡처 기능을 [설정]으로 변경합니다.



3. 다음과 같이 구성한 후 [변경 내용 저장]을 클릭합니다.
 - 시간 범위(Time windows): 1분으로 선택합니다. 이는 덤프를 생성할 시간 간격을 지정합니다.
 - 창 크기(Size windows): 100MB로 선택합니다. 데이터 파티션의 크기가 지정한 값을 초과하면 blob 스토리지에 새 blob을 작성합니다.
 - 캡처 시간 동안 이벤트가 발생하지 않으면 빈 파일을 내보내지 않음: 이 옵션을 선택하지 않으면 위에서 설정한 시간 범위 동안 이벤트가 없더라도 blob 스토리지에 파일을 생성합니다.
 - 공급자 캡처(Capture provider): Azure 스토리지 계정과 Data Lake Store Gen 1 중 하나를 선택할 수 있습니다. "Azure Storage 계정"을 선택합니다.
 - Azure Storage 컨테이너: [컨테이너 선택]을 클릭한 후 앞서 만들었던 스토리지 계정의 "capture" 컨테이너를 선택합니다.
 - 샘플 캡처 파일 이름 형식 / 캡처 파일 이름 형식: blob에 생성될 폴더 이름 지정 패턴을 선택할 수 있습니다.

변경 내용 저장 취소

Azure Event Hubs 캡처를 사용하면 Event Hubs의 스트리밍 데이터를 원하는 Azure Blob Storage 또는 Azure Data Lake Store 계정에 자동으로 전달할 수 있으며, 시간 또는 크기 간격도 원하는 대로 유연하게 설정할 수 있습니다. 캡처 설정은 빠르고 간편하며, 실행하는 데 드는 관리 비용이 없고, Event Hubs 처리량 단위에 맞춰 자동으로 스케일링됩니다. Event Hubs 캡처는 스트리밍 데이터를 Azure에 로드하는 가장 쉬운 방법이며, 사용자가 데이터 캡처보다 데이터 처리에 중점을 둘 수 있도록 지원합니다. 캡처에 대해 자세히 알아보세요.

캡처
 설정 해제

참고: 캡처를 사용하면 이 계정에 추가 요금이 부과됩니다. 가격 책정에 대해 자세히 알아보세요.

시간 범위(분) 1

창 크기(MB) 100

☐ 캡처 시간 동안 이벤트가 발생하지 않으면 빈 파일을 내보내지 않음

공급자 캡처
 Azure Storage 계정

Azure Storage 컨테이너 *
 capture 컨테이너 선택

스토리지 계정
 /subscriptions/6252362-810-438-8679-65c3de1280a/resourceGroups/81_jhwooeventhubs/providers/Microsoft.Storage/storageAccounts/jhwoochekpoint

샘플 캡처 파일 이름 형식
 {Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

캡처 파일 이름 형식 ①
 {Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

e.g. jhwooeventhubs/my-demo/0/2022/02/02/12/44/29

4. 위에서 지정한 시간 범위를 기다린 후 Azure 스토리지 계정 블레이드로 이동합니다. 위에서 지정한 컨테이너로 이동하면 아래와 같이 Event Hub 네임스페이스 이름의 캡처가 생성된 것을 확인할 수 있습니다. 이 컨테이너를 클릭하면 지정한 캡처 파일 이름 형식에 따라 폴더가 생성된 것을 확인할 수 있습니다.

capture ...

컨테이너

검색(Ctrl+/) << 업로드 액세스 수준 변경 새로 고침 삭제 계층 변경 임대 가져오기 ...

개요 인증 방법: 액세스 키 (Azure AD 사용자 계정으로 전환)
 위치: capture

접두사로 Blob 검색(대/소문자 구분)
☐ 삭제된 BLOB 표시

필더 추가

이름	수정한 날짜	액세스 계층	보관 상태
jhwooeventhubs			

메타데이터
 편집기(미리 보기)