

한국 마이크로소프트

Microsoft Technical Trainer

Enterprise Skills Initiative

Microsoft Azure

Azure Event Grid

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 02월 03일 한국 마이크로소프트(유), Microsoft Technical Trainer 팀 우진환에 의해 TASK 01 ~ TASK 02부분이 작성되었습니다. (Version 0.7.0)
2. 이 문서는 2022년 02월 05일 한국 마이크로소프트(유), Microsoft Technical Trainer 팀 우진환에 의해 TASK 03 부분이 추가되었습니다. (Version 1.0.0)

목차

AZURE 메시징 서비스 소개.....	5
AZURE EVENT GRID.....	5
AZURE EVENT HUBS	6
AZURE SERVICE BUS	6
AZURE 메시징 서비스 비교.....	8
EVENT GRID와 EVENT HUBS의 차이점.....	8
EVENT GRID 사용 시나리오	8
EVENT GRID 이점.....	9
TASK 01. 빌트-인 이벤트 사용(CONSUME)	10
TASK 02. 커스텀 이벤트(CUSTOM EVENT) 전송	21
TASK 03. 커스텀 이벤트(CUSTOM EVENT)를 구독(SUBSCRIBE).....	30

Azure 메시징 서비스 소개

이벤트(event)를 전달하는 서비스와 메시지(message)를 전달하는 서비스 간에는 중요한 차이점이 있습니다.

- 이벤트(event)는 조건 혹은 상태 변경에 대한 간단한 알림입니다. 이벤트 **publisher**는 이벤트가 처리되는 방식에 대해 기대하지 않습니다. 이벤트 **consumer**는 알림을 어떻게 처리할지 결정합니다. 이벤트를 개별 단위 혹은 시리즈의 일부일 수 있습니다. 예를 들어 이벤트는 파일이 생성되었다는 것을 **consumer**에게 알립니다. 이 경우 파일에 대한 일반 정보가 포함될 수 있지만 파일 자체는 없습니다.
- 메시지(message)는 다른 곳에서 사용하거나 저장하기 위해 서비스에서 생성한 원시 데이터입니다. 메시지는 메시지 파이프라인을 트리거한 데이터가 포함되어 있습니다. 메시지 **publisher**는 **consumer**가 메시지를 처리하는 방법을 기대합니다. 따라서 양측 사이에 일종의 계약이 존재합니다. 예를 들어 **publisher**는 원시 데이터가 포함된 메시지를 보내고 **consumer**가 해당 데이터에서 파일을 만들고 작업이 완료되면 응답을 보낼 것으로 기대합니다.

Azure Event Grid와 Azure Event Hub는 이벤트 기반 아키텍처를 사용하는 애플리케이션을 빌드하는데 사용되는 서비스입니다. 이벤트 기반 아키텍처는 이벤트를 통해 정보를 공유하여 서로 다른 시스템과 커뮤니케이션하는 애플리케이션을 구축할 수 있는 기능을 제공합니다.

Azure Event Grid

Event Grid는 Azure Blog 스토리어나 Azure 미디어 서비스와 같은 다양한 소스의 이벤트를 Azure Functions나 Webhook과 같은 다양한 핸들러로 배포할 수 있습니다.



Event Grid에서 사용되는 용어는 아래와 같습니다.

- Event (일어난 일에 대한 설명): 이벤트는 조건 혹은 상태 변경에 대한 경량의 알림이며, 변경된 개체 전체를 포함하지 않습니다. 이벤트 전송자는 **publisher**라고 하고 받는 쪽을 **subscriber**라고 합니다.
- Event Source (이벤트가 발생한 장소): 이벤트 소스(publisher)는 Event Grid에 이벤트를 보내는 역할을 합니다. 각 Event Source는 하나 이상의 이벤트 유형과 관련됩니다. 예를 들어 Azure Storage는 blob 생성 이벤트에 대한 이벤트 소스입니다.
- Topic (publisher가 이벤트를 전송하는 엔드포인트): 이벤트 토픽은 이벤트를 그룹으로 분류합니다. 이벤트 소스는 공용 엔드포인트를 사용하여 토픽에 이벤트를 전송합니다. 토픽에는 2가지 유형이 있습니다. 시스템 토픽(**system topic**)은 Azure 서비스에서 제공하는 빌트-인 토픽이며, Azure 구독에서 시스템 토픽을 볼 수 없지만 **publisher**가 토픽을 소유하고 있기 때문에 구독(**subscribe**)할 수는 있습니다. 커스텀 토픽(**custom topic**)은 애플리케이션 및 3rd party 토픽입니다. 커스텀 토픽을 생성하거나 액세스를 할당하면 구독에서 커스텀 토픽을 볼 수 있습니다.

- Event subscription (이벤트를 라우팅하기 위한 엔드포인트 혹은 빌트-인 메커니즘이며, 종종 여러 핸들러로 라우팅): 이벤트 구독은 이벤트 핸들러가 수신하려는 토픽의 이벤트를 정의합니다. 구독은 유형이나 주제에 따라 이벤트를 필터링할 수 있으며 이벤트 핸들러가 관련 이벤트만 수신하도록 할 수 있습니다.
- Event Handler (이벤트를 수신하는 애플리케이션 혹은 서비스): 이벤트 핸들러(subscriber)는 Event Grid에서 이벤트를 수신할 수 있는 모든 구성 요소입니다.

Azure Event Hubs

Azure Event Hubs는 게시-구독(publish-subscribe) 커뮤니케이션 패턴을 위한 중개자입니다. Event Grid와 달리 Event Hub는 낮은 대기 시간으로 대용량의 이벤트(초당 수백만개의 이벤트)를 처리하는 서비스입니다. Event Hub를 이벤트 처리 파이프라인의 시작점으로 고려할 수 있습니다. 또한 Event Hub를 Event Grid 서비스의 이벤트 소스로 사용할 수도 있습니다.

Event Hub에는 여러 파티션(partition)이 포함되어 있습니다. Event Hub는 데이터를 수신하고 이를 파티션으로 나눕니다. 파티션은 데이터가 저장되는 버퍼입니다. 이러한 버퍼로 인해 subscriber가 바쁘거나 오프라인이라는 이유로 이벤트를 놓치지 않습니다. subscriber는 이벤트를 가져오기 위해 이 버퍼를 항상 사용합니다. 기본적으로 이벤트는 자동으로 만료되기 전에 이 버퍼(파티션)에 지정한 시간(예를 들어 24시간) 동안 남아 있습니다. 이러한 버퍼는 데이터가 서로 나누어지기 때문에 파티션이라고 합니다.

Event Hub의 장점 중 하나는 약간의 구성 변경으로 기존 Kafka에서 사용할 수 있는 Kafka 엔드포인트를 제공한다는 것입니다. Kafka와 Event Hub의 가장 큰 차이점은 Event Hub가 클라우드 서비스라는 것이며, 이는 관리할 서버나 네트워크가 없다는 것을 의미합니다. 또 다른 장점으로 Event Hub는 로깅 및 원격 분석에 사용할 수 있다는 것입니다. 또한 Stream Analytics (serverless real-time 분석) 및 Power BI (비즈니스 분석 서비스)와 통합할 수 있습니다.

Azure Service Bus

Azure Service Bus는 메시지 큐와 게시-구독 토픽(publish-subscribe topic)이 있는 완전히 관리되는 엔터프라이즈 메시지 브로커입니다. 이 서비스는 트랜잭션, 주문, 중복 탐지 및 즉각적인 일관성이 필요한 엔터프라이즈 애플리케이션을 위한 것입니다. Service Bus는 클라우드 네이티브 애플리케이션에 비즈니스 프로세스에 대한 안정적인 상태 전환 관리를 제공할 수 있습니다. 손실되거나 복제할 수 없는 중요한 메시지를 처리할 때 Azure Service Bus를 사용하는 것이 좋습니다. Azure Service Bus는 하이브리드 클라우드 솔루션 전반에서 매우 안전한 커뮤니케이션을 가능하게 하고 기존 온-프레미스 시스템을 클라우드 솔루션에 연결할 수 있습니다.

Service Bus는 브로커된 메시징 시스템(brokered messaging system)입니다. Service Bus는 소비 당사자가 메시지를 받을 준비가 될 때까지 브로커(broker; 예를 들어 큐)에 메시지를 저장하며 다음과 같은 특징이 있습니다.

- 폴링(polling)이 필요한 신뢰할 수 있는 비동기식 메시지 전달(enterprise messaging as a service)
- 고급 메시징 기능. 예를 들어 FIFO (first-in and first-out), 일괄 처리/세션, 트랜잭션, dead-lettering, 임시 제어, 라우팅 및 필터링, 중복 탐지

- 하나 이상의 메시지 전달
- 메시지의 순차적인 전달(선택 사항)

Azure 메시징 서비스 비교

Event Grid와 Event Hubs의 차이점

Event Grid와 Event Hub의 가장 큰 차이점은 Event Hubs는 데이터 수집을 위한 엔드포인트만 수락하고 **publisher**에게 데이터를 다시 보내는 메커니즘을 제공하지 않는다는 것입니다. 반면 Event Grid는 **publisher**에서 발생하는 이벤트를 알리기 위해 HTTP 요청을 보냅니다.

- Event Grid는 Azure Function을 트리거할 수 있습니다. Event Hubs의 경우 Azure Functions는 이벤트를 가져와 처리해야 합니다.
- Event Grid는 큐 매커니즘(queueing mechanism)이 아닌 분배 시스템입니다. 이벤트가 푸시되면 즉시 처리(push out)되고, 배달되지 않은 이벤트를 스토리지 계정으로 전송하지 않는 이상 처리되지 않은 이벤트는 사라집니다. 이 프로세스를 dead-lettering이라고 합니다. 기본적으로 이 옵션은 비활성화되어 있으며 이를 활성화하고자 하는 경우 Event Grid를 만들 때 스토리지 계정을 지정해야 합니다.
- Event Hubs에서 **publisher**와 **subscriber**는 모두 내구성이 있는 저장소에서 읽고 씁니다. 데이터는 Event Hubs에 최대 90일(프리미엄 티어) 동안 보관할 수 있습니다. 이를 통해 특정 시점에서 다시 시작하거나 이전 시점에서 다시 시작하고 필요할 때 이벤트를 다시 처리할 수 있습니다.
- Event Grid는 이벤트의 순서를 보장하지 않습니다. 반대로 Event Hubs는 파티션을 사용하며 이러한 파티션은 순서가 지정된 시퀀스이므로 동일한 파티션에서 이벤트의 순서를 유지할 수 있습니다.

결론적으로 Event Hubs는 초당 수백만개의 이벤트를 수신 및 처리하고 대기 시간이 짧은 이벤트 처리를 제공하는 서비스가 필요할 때 더 적합한 솔루션입니다. 또한 Event Hubs는 동시에 여러 소스의 데이터를 처리하고 다양한 스트림 처리 인프라 및 분석 서비스로 이벤트를 라우팅할 수 있습니다. 따라서 Event Hubs는 원격 분석 시나리오에서 더 많이 사용됩니다.

반면 Event Grid는 항목이 배송되거나 스토리지에 항목이 추가 및 업데이트되는 경우와 같은 반응형 시나리오(reactive scenario)에 더 이상적입니다. 또한 Azure Functions, Logic Apps, Webhook과 같은 기본 Azure 서비스와 통합도 제공합니다. 따라서 Event Grid는 Event Hubs 보다 저렴하며 빅 데이터를 처리할 필요가 없을 때 더 적합합니다.

Event Grid 사용 시나리오

- Event Grid의 가장 일반적인 시나리오 중 하나는 serverless 애플리케이션 아키텍처입니다. 따라서 Blob Storage와 같은 이벤트에 거의 즉각적으로 반응할 수 있으며 새 이미지가 스토리지에 업로드될 때마다 Event Grid를 사용하여 반응하고 해당 이미지를 즉시 처리할 수 있습니다.
- 운영 자동화에 사용할 수 있습니다. Event Grid는 리소스 그룹 및 기타 구독의 이벤트에 반응할 수 있기 때문에 Azure Automation과 같은 서비스에 알림을 보내 진단을 실행하거나 Azure 인프라에서 자동화 스크립트를 실행할 수 있습니다.
- 애플리케이션 통합을 수행할 수 있습니다. Event Grid 토픽(topic)에서 커스텀 엔드포인트를 만들 수 있기 때문에 이 커스텀 이벤트를 애플리케이션에 전송할 수 있습니다. 즉 Event Grid 토픽으로 다른 서비스와 애플리케이션을 연결할 수 있습니다.

Event Grid 이점

- 단순함: 몇 번의 클릭으로 Azure 리소스의 이벤트를 이벤트 핸들러나 엔드포인트를 가리키도록 구성할 수 있습니다.
- 고급 필터링: 이벤트 핸들러가 관련 이벤트만 수신하도록 하기 위해 이벤트 유형이나 이벤트 게시 경로를 필터링할 수 있습니다.
- Fan-out: 하나의 이벤트가 발생할 때 여러 subscriber가 동일한 이벤트를 수신하도록 할 수 있습니다. 즉 여러 엔드포인트가 동일한 이벤트를 구독하도록 하여 이벤트의 복사본을 원하는 모든 곳에 전송할 수 있습니다.
- 신뢰성: 서비스가 이벤트 처리에 실패하면 Event Grid는 exponential backoff로 24시간 동안 해당 이벤트를 계속 보내려고 시도합니다.

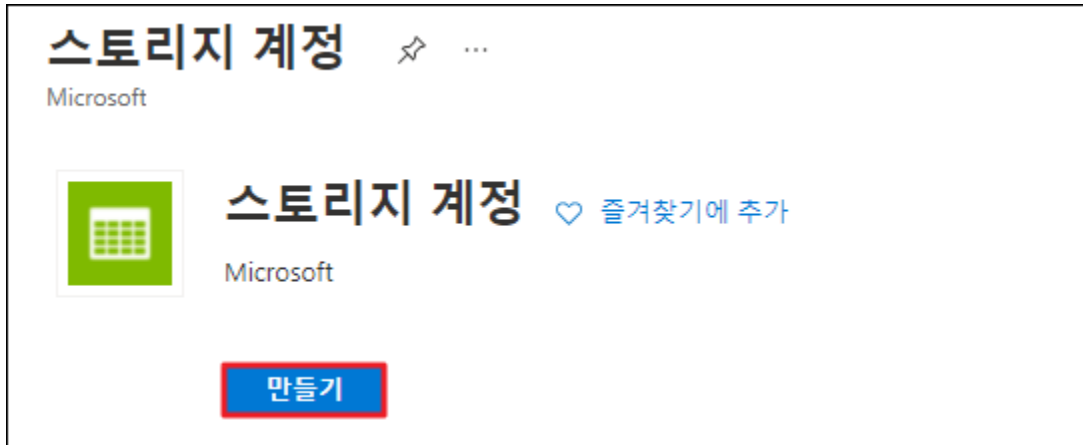
위에서 설명한 3개의 메시징 서비스를 비교하면 아래와 같습니다.

서비스	목적	유형	사용 시기
Event Grid	반응형 프로그래밍	이벤트 배포(불연속)	상태 변경에 대한 반응
Event Hubs	빅 데이터 파이프라인	이벤트 스트리밍(연속)	원격 분석 및 분산 데이터 스트리밍
Service Bus	높은 가치의 엔터프라이즈 메시징	메시지	주문 처리 및 금융 거래

TASK 01. 빌트-인 이벤트 사용(consume)

이 작업에서는 Azure에서 기본적으로 제공되는 시스템 토픽을 사용하여 스토리지 계정에 대한 이벤트를 생성합니다.

1. Azure 포털에서 [리소스 만들기]를 클릭한 후 "스토리지 계정"을 검색하고 클릭합니다. [스토리지 계정] 블레이드에서 [만들기]를 클릭합니다.



2. [저장소 계정 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.
 - [프로젝트 정보 - 리소스 그룹]: "새로 만들기"를 클릭한 후 "03_eventGridRg"를 입력합니다.
 - [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 * (신규) 03_eventGridRg

[새로 만들기](#)

인스턴스 정보

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

스토리지 계정 이름 * jhwooeventgrid

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)

3. 새로 만든 스토리지 계정 블레이드로 이동합니다. [스토리지 계정] 블레이드의 [데이터 스토리지 - 컨테이너]로 이동한 후 메뉴에서 [컨테이너]를 클릭합니다. [새 컨테이너]에서 컨테이너 이름에 "demo"를 입력한 후 [만들기]를 클릭합니다.

홈 > 리소스 그룹 > 03_eventGridRg > jhwooeventgrid

jhwooeventgrid | 컨테이너 ...

스토리지 계정 | 디렉터리: Microsoft

검색(Ctrl+/) << + 컨테이너 액세스 수준 변경 컨테이너 복원 >>

데이터 스토리지

컨테이너

파일 공유

큐

테이블

보안 + 네트워킹

접두사로 컨테이너 검색

☒ 삭제된 컨테이너 표시

이름	마지막으로 수정한 ...	공용
\$logs	2022. 2. 4. 오후 4:55:...	프...

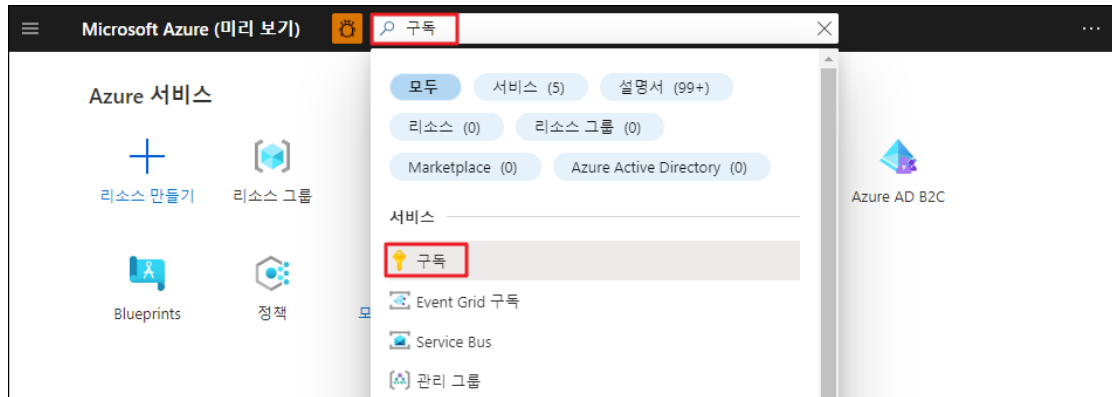
새 컨테이너 ✕

이름 * demo ✓

공용 액세스 수준 * 프라이빗(익명 액세스 없음) ✓

고급

4. 다른 일반적인 Azure 서비스와 달리 Event Grid는 [리소스 만들기]를 통해 추가하지 않습니다. Event Grid는 Azure의 기본 필수 서비스이기 때문에 Event Grid를 사용하기 위해서는 토픽(topic)과 구독(subscription)만 만들면 됩니다. Event Grid를 사용하기 위해서는 Azure 리소스 공급자에서 Event Grid를 활성화해야 합니다. Azure 포털의 검색창에서 "구독"을 검색한 후 클릭합니다.

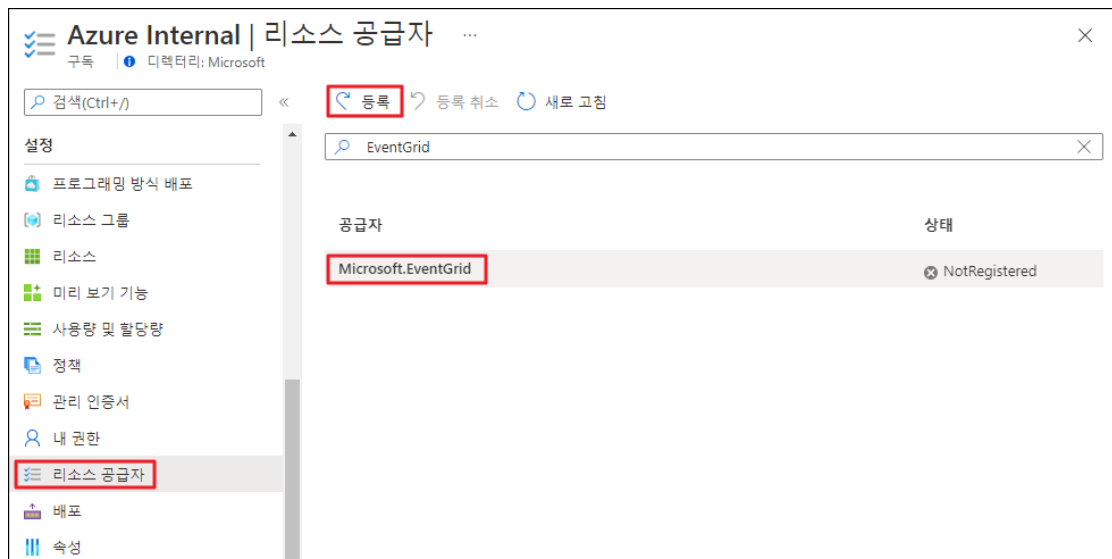


5. [구독] 블레이드에서 자신의 구독을 클릭합니다.

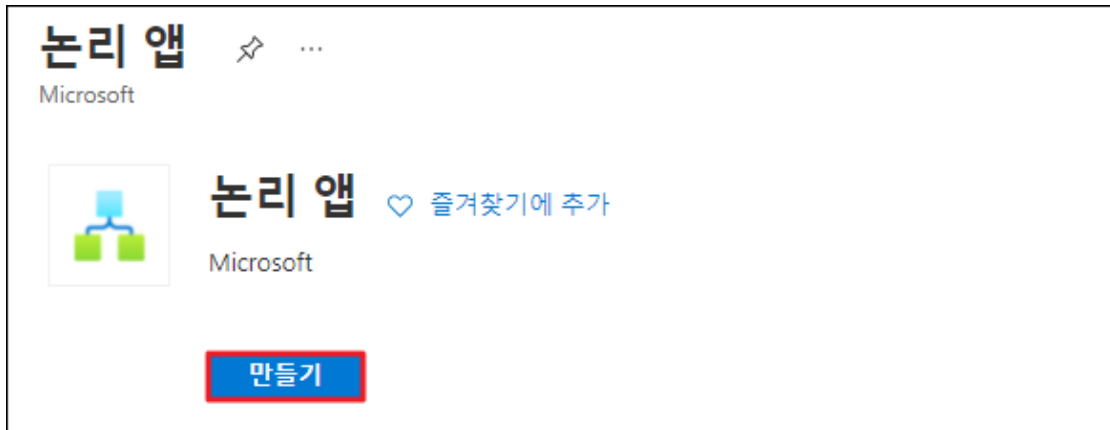


6. [구독] 블레이드의 [설정 - 리소스 공급자]로 이동한 후 "EventGrid"를 검색합니다.

"Microsoft.EventGrid" 리소스 공급자를 선택한 후 메뉴에서 [등록]을 클릭합니다. 페이지를 새로 고쳐 리소스 공급자 등록이 완료될 때까지 기다립니다.



7. 이 첫 번째 실습에서는 시스템 토픽을 사용할 것이기 때문에 별도의 커스텀 토픽(topic)을 만들 필요 없이 Logic App에서 바로 Event Grid를 사용할 수 있습니다. Azure 포털에서 [리소스 만들기]를 클릭한 후 "논리 앱"을 검색합니다. [논리 앱] 블레이드에서 [만들기]를 클릭합니다.



8. [논리 앱 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.

- [프로젝트 세부 정보 - 리소스 그룹]: 03_eventGridRg
- [인스턴스 정보 - 유형]: 소비
- [인스턴스 정보 - 논리 앱 이름]: NewBlobEvent
- [인스턴스 정보 - 지역]: Korea Central
- [인스턴스 정보 - Log Analytics 사용 설정]: 아니요

논리 앱 만들기

기본 태그 검토 + 만들기

순쉬운 리소스 관리, 배포 및 공유를 위해 논리적 단위로 워크플로를 그룹화할 수 있는 논리 앱을 만듭니다. 워크플로를 사용하면 업무상 중요한 앱 및 서비스를 Azure Logic Apps에 연결하고 코드를 한 줄도 작성하지 않고 워크플로를 자동화할 수 있습니다.

프로젝트 세부 정보

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

구독 * ⓘ Azure Internal ▼

리소스 그룹 * ⓘ 03_eventGridRg ▼
[새로 만들기](#)

인스턴스 정보

유형 * ☒ 소비 ☐ 표준
 ⓘ 클래식 소비 경험 만들기를 찾고 계십니까? [여기를 클릭](#)

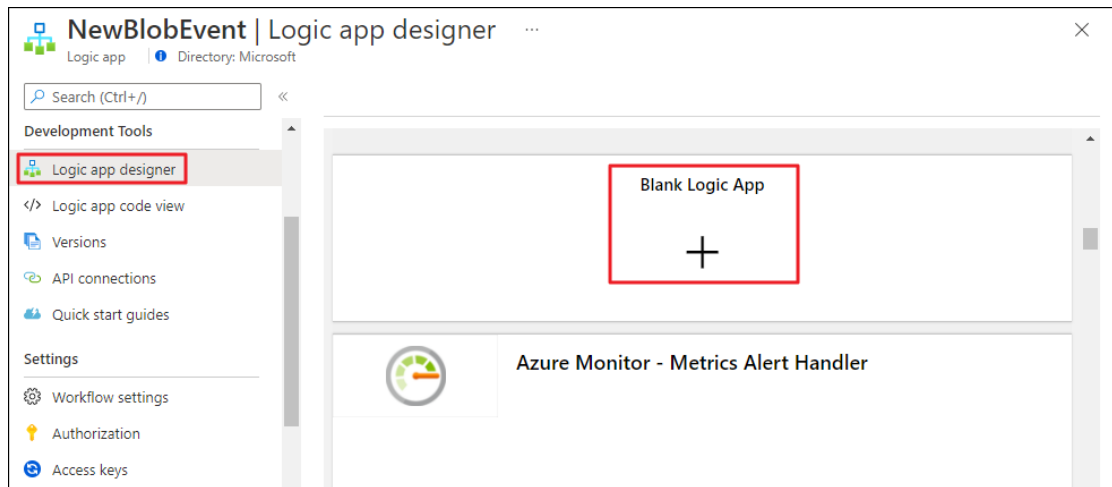
논리 앱 이름 * NewBlobEvent ✓

지역 * Korea Central ▼

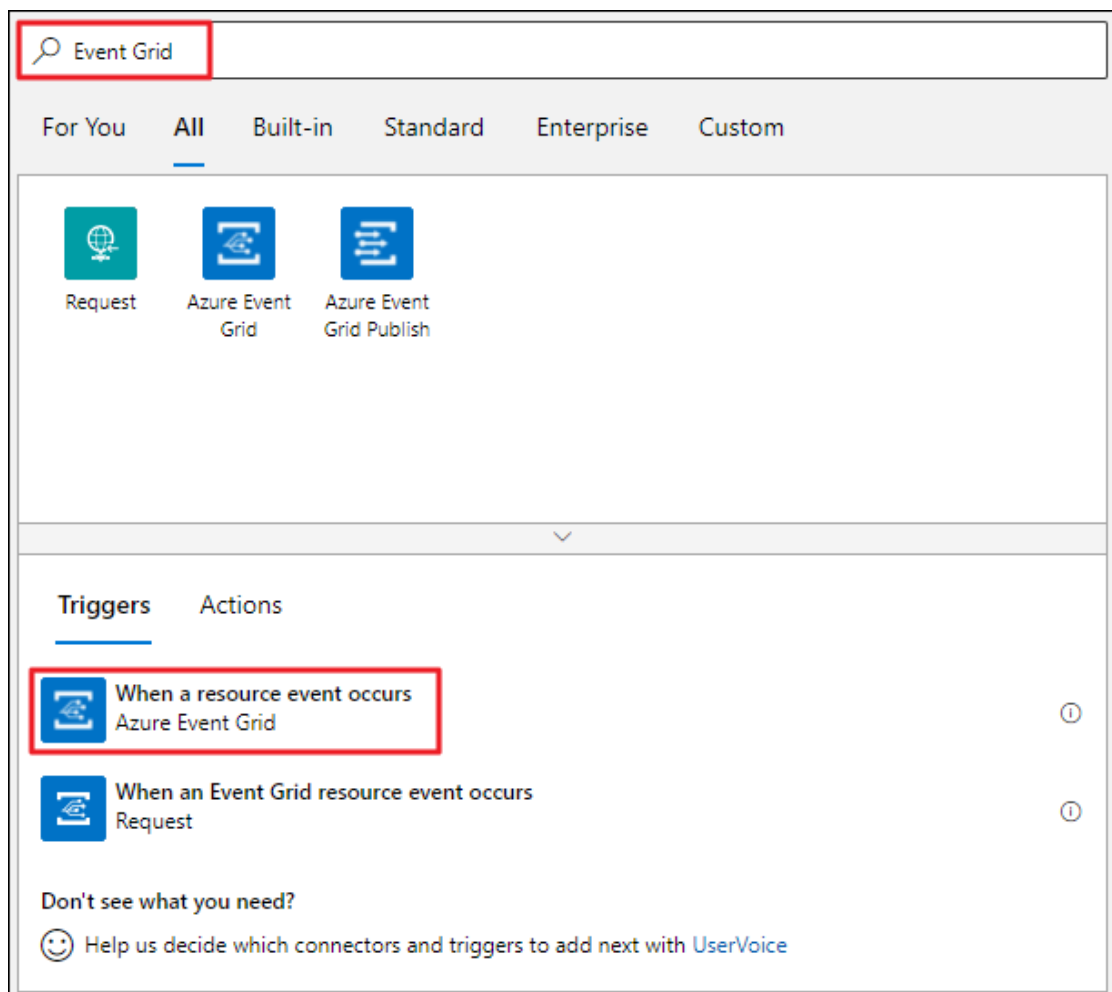
Log analytics 사용 설정 * ☐ 예 ☒ 아니요

⚠ 선택한 구독에 Log analytics 작업 영역 리소스가 없습니다. Log analytics를 사용 설정하려면 새 Log analytics 작업 영역 리소스를 만들거나 이미 있는 구독으로 전환하세요.

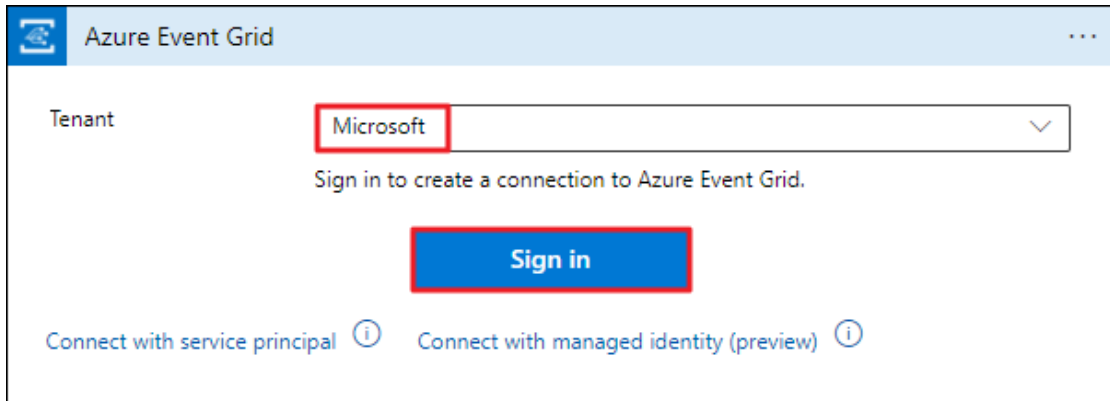
9. Logic App 설정을 위해 Azure 포털의 언어를 영어로 변경합니다. 한국어에서 Logic App의 커넥터 및 트리거 검색이 용이하지 않습니다. 새로 만든 [NewBlobEvent Logic app] 블레이드의 [Development Tools - Logic app designer]로 이동합니다. [Blank Logic App] 타일을 클릭합니다.



10. 디자이너의 검색창에서 "Event Grid"를 검색한 후 "When a resource event occurs" 트리거를 클릭합니다.

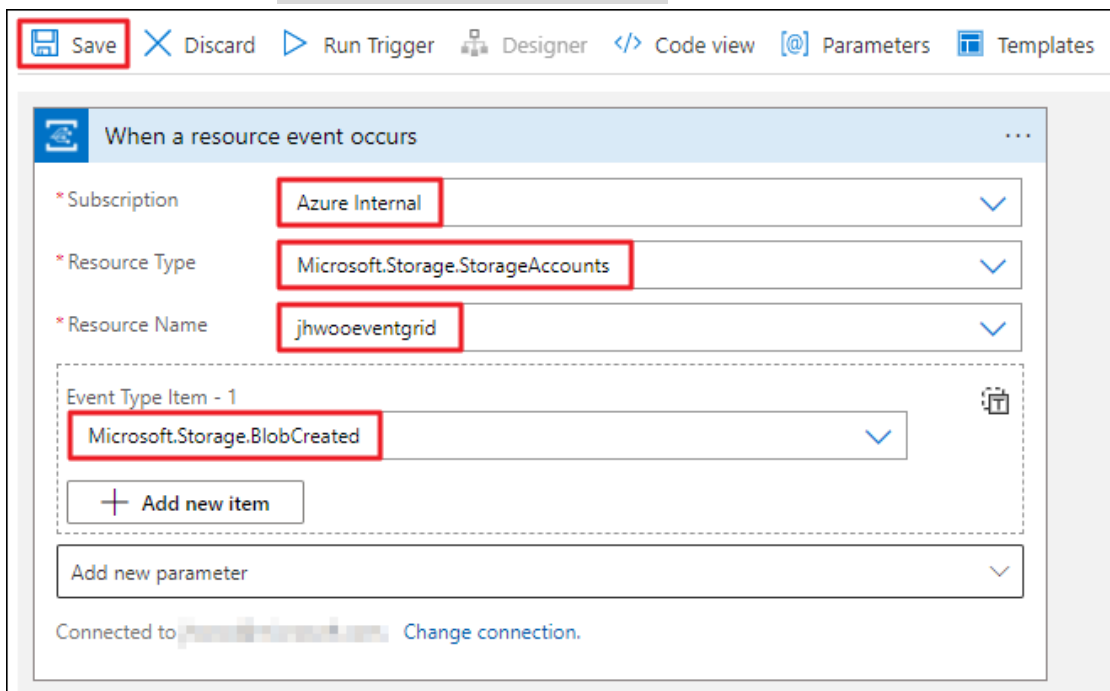


11. [Azure Event Grid] 타일에서 [Sign in]을 클릭하여 자신의 구독에 로그인합니다.



12. [When a resource event occurs] 타일에서 아래와 같이 구성한 후 상단 메뉴에서 [Save]를 클릭하여 Logic App을 저장합니다.

- Subscription: 자신의 Azure 구독을 선택합니다.
- Resource Type: `Microsoft.Storage.StorageAccounts`
- Resource Name: 앞서 만들었던 스토리지 계정을 선택합니다.
- Event Type Item - 1: `Microsoft.Storage.BlobCreated`



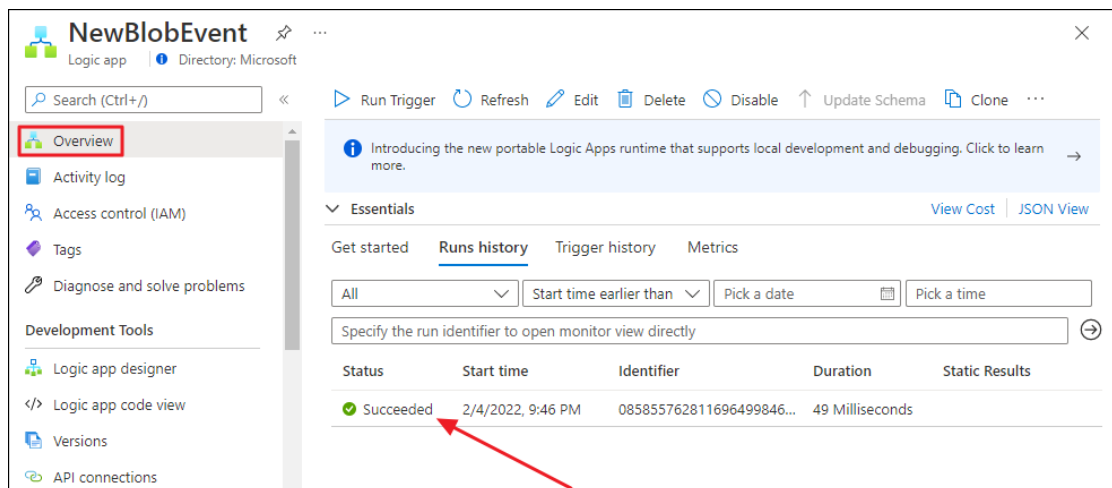
13. 테스트를 위해 앞서 만들었던 [스토리지 계정] 블레이드의 [데이터 스토리지 - 컨테이너]로 이동한 후 demo 컨테이너를 클릭합니다.



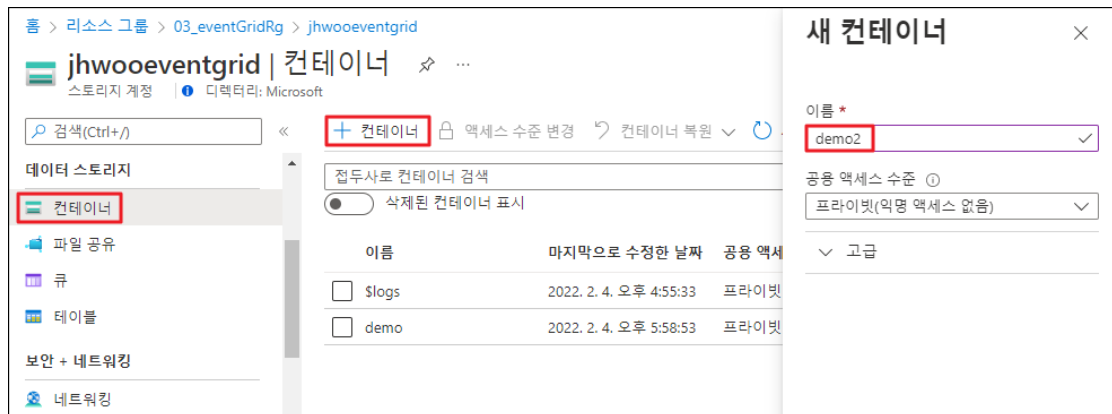
14. [demo 컨테이너] 블레이드의 메뉴에서 [업로드]를 클릭합니다. [Blob 업로드]에서 [찾아보기]를 클릭한 후 아무 파일이나 선택한 후 [업로드]를 클릭합니다.



15. [NewBlobEvent Logic app] 블레이드의 [Overview]로 이동합니다. Event Grid는 Blob 스토리지에 파일이 업로드되는 순간 실행되고 실행 내용이 표시되는 것을 확인할 수 있습니다. 즉 Event Grid는 이러한 실시간 애플리케이션을 위해 디자인되었습니다.



16. 다시 [스토리지 계정] 블레이드로 이동한 후 [데이터 스토리지 - 컨테이너]에서 [컨테이너]를 클릭합니다. [새 컨테이너]에서 이름에 "demo2"를 입력한 후 [만들기]를 클릭합니다.

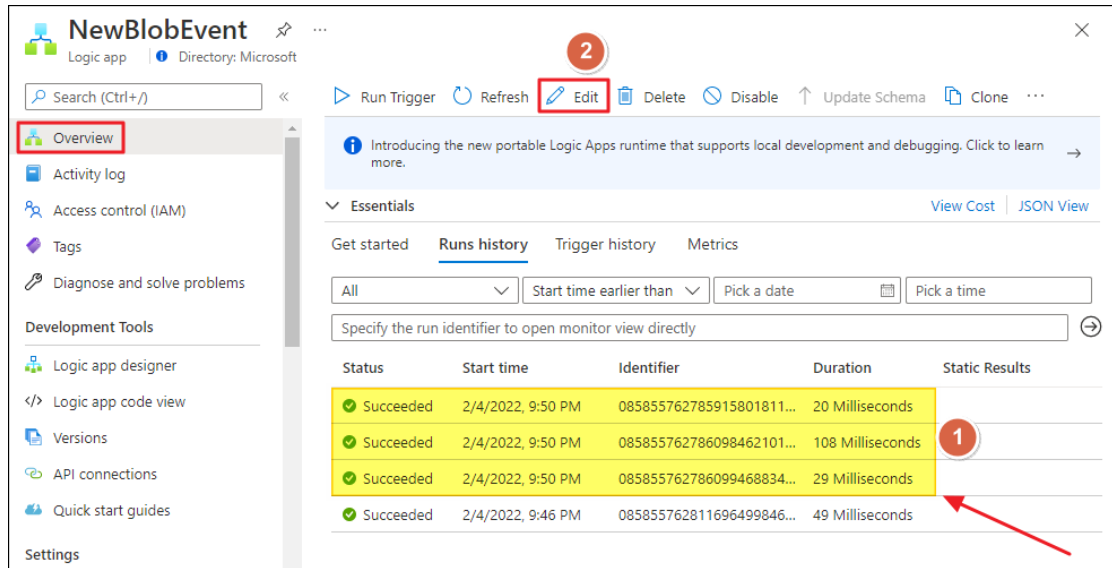


17. 새로 만든 demo2 컨테이너로 이동한 후 메뉴에서 [업로드]를 클릭합니다. [Blob 업로드]에서 [찾아보기]를 클릭한 후 아무 파일(이 예제에서는 3개의 임의의 파일)을 선택하고 [업로드]를 클릭합니다.

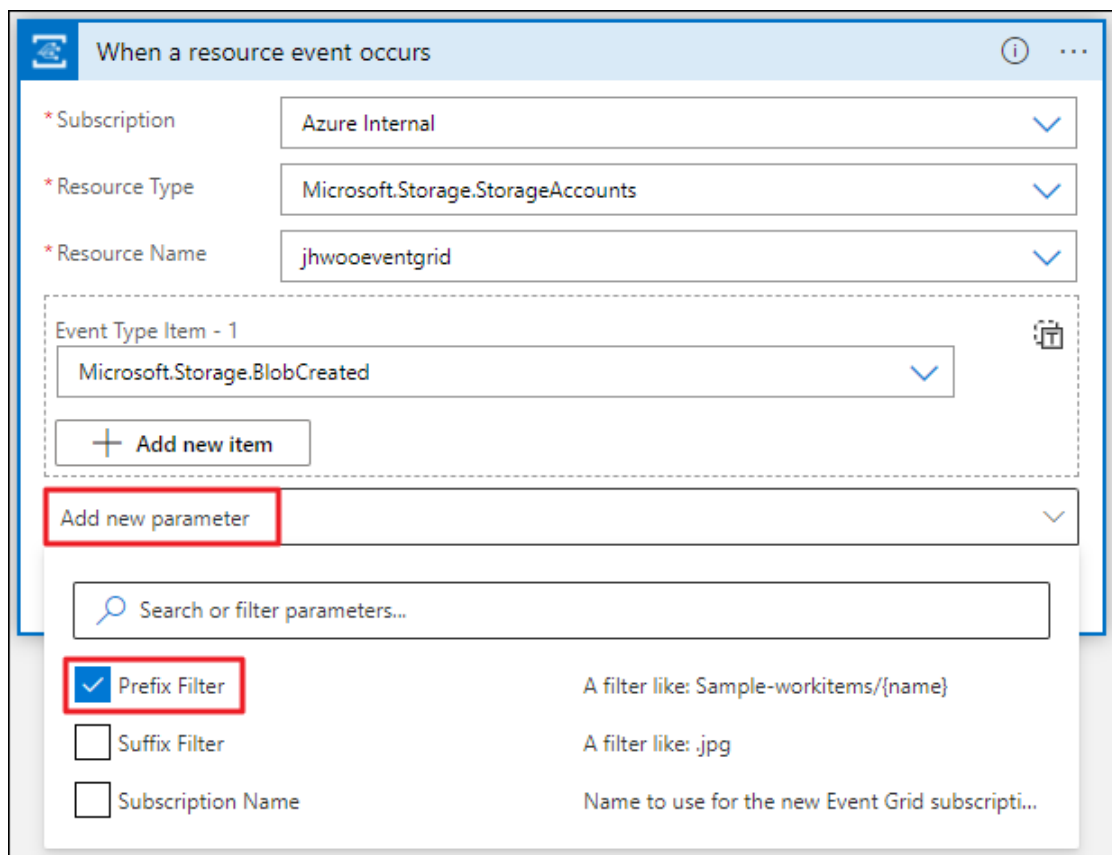


18. [NewBlobEvent] Logic app] 블레이드의 [Overview]로 이동한 후 메뉴에서 [Refresh]를 클릭합니다. 아래와 같이 스토리지 계정에 새로 만든 컨테이너에 업로드한 파일에 대해서도 Event Grid가 실행되는 것을 확인할 수 있습니다. 아래 내용을 확인한 후 [NewBlobEvent] Logic app] 블레이드의 [Overview]에서 [Edit]를 클릭합니다.

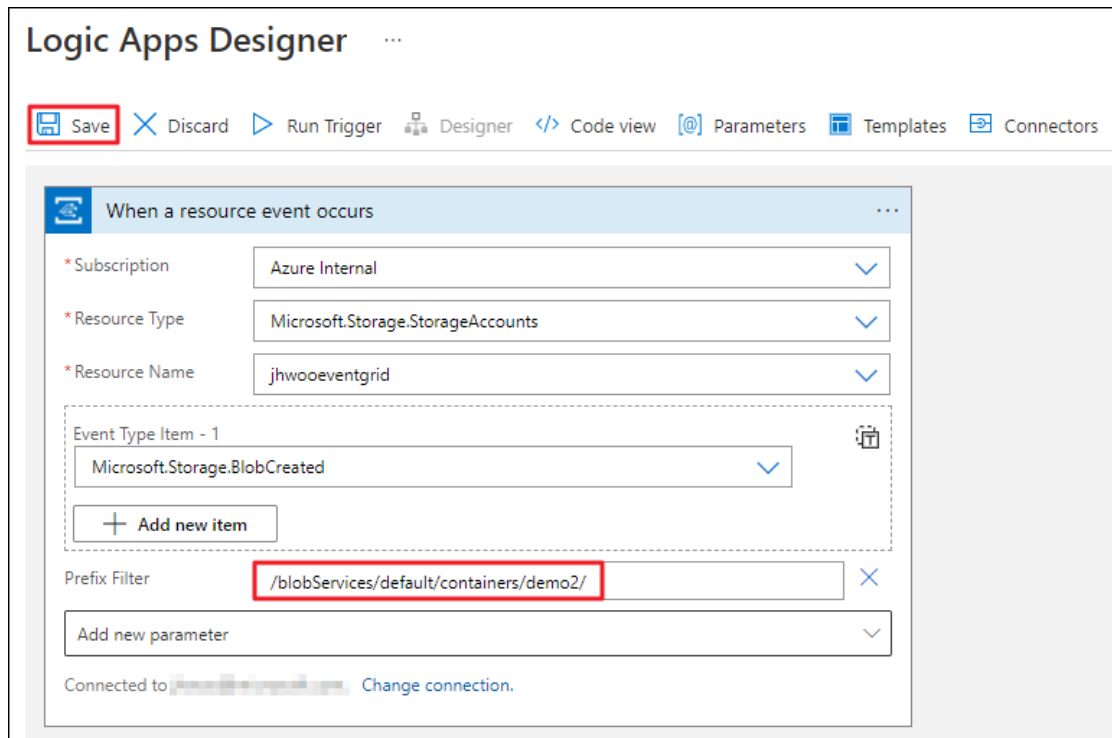
- 이 Event Grid의 좋은 기능 중 하나이지만 Event Grid를 디자인할 때 고려해야 할 기능이기도 합니다.
- 만약 특정 컨테이너에 blob이 업로드 되었을 때만 Event Grid를 실행하고자 한다면 필터를 사용하여 단일 컨테이너의 이벤트만 필터링해야 합니다.



19. Logic App 디자이너에서 [When a resource event occurs] 타일을 클릭한 후 "Add new parameter" 드롭다운 메뉴를 클릭하고 "Prefix Filter"를 체크합니다.



20. [When a resource event occurs] 타일의 "Prefix Filter"에 `/blobServices/default/containers/demo2/`를 입력한 후 메뉴에서 [Save]를 클릭합니다. 이제 이 Event Grid는 `demo2`라는 컨테이너의 이벤트에만 반응하게 됩니다.



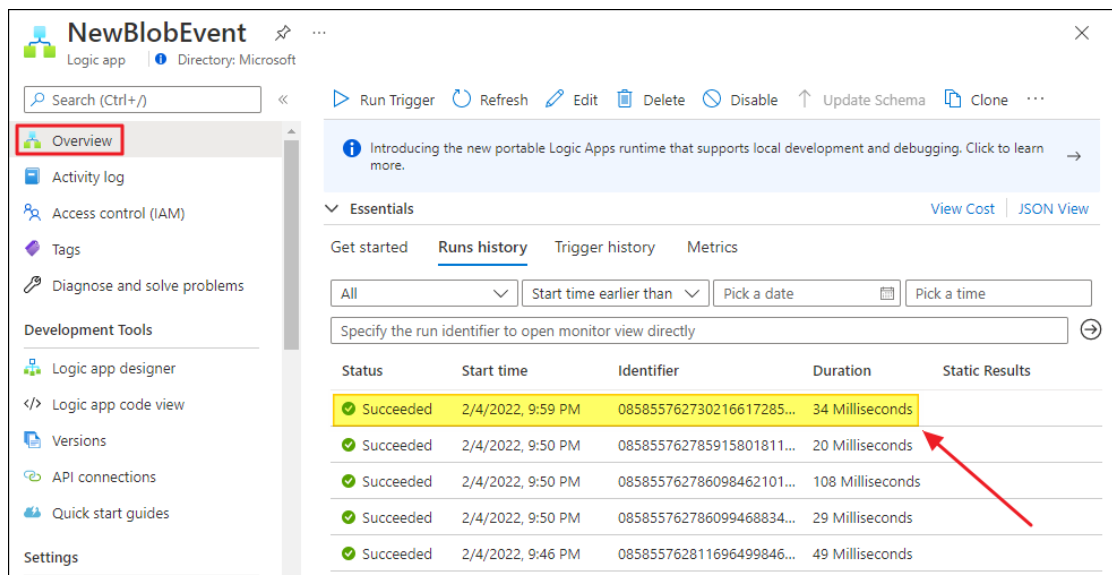
21. [스토리지 계정] 블레이드로 이동한 후 앞서 만들었던 [demo2 컨테이너] 블레이드로 이동합니다. [업로드]를 클릭한 후 임의의 파일 1개를 업로드합니다.



22. [demo 컨테이너] 블레이드로 이동한 후 [업로드]를 클릭하고 임의의 파일 3개를 업로드합니다.



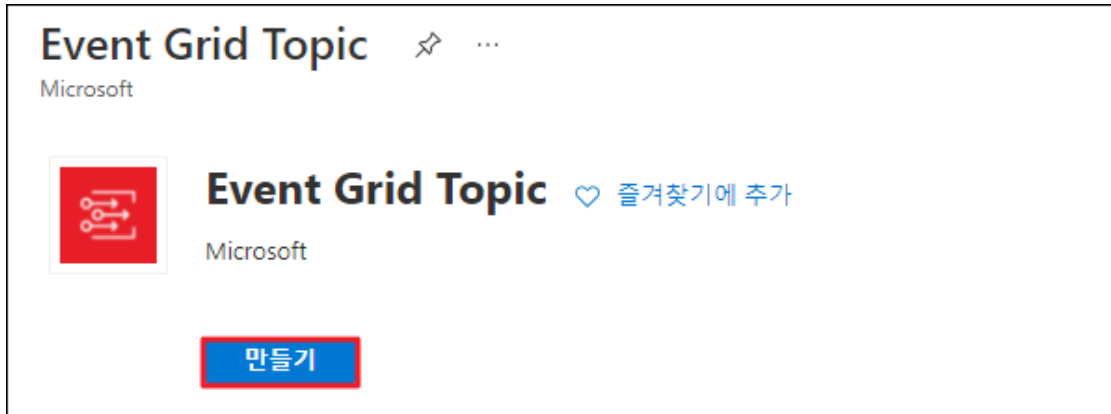
23. 다시 [NewBlobEvent Logic app] 블레이드의 [Overview]로 이동한 후 [Refresh]를 클릭합니다. **demo2** 컨테이너에 1개의 파일을 업로드했고 **demo** 컨테이너에 3개의 파일을 업로드하였지만 Logic App에서 **demo2** 컨테이너에만 반응하도록 필터를 설정하였기 때문에 아래와 같이 1개의 실행 히스토리만 표시되는 것을 확인할 수 있습니다.



TASK 02. 커스텀 이벤트(custom event) 전송

이 실습에서는 커스텀 이벤트를 만들고 테스트합니다.

1. Azure 포털에서 [리소스 만들기]를 클릭한 후 "Event Grid"를 검색하고 "Event Grid Topic"을 클릭합니다.
[Event Grid Topic] 블레이드에서 [만들기]를 클릭합니다.



2. [토픽 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성한 후 [다음]을 클릭합니다.
 - [프로젝트 세부 정보 - 리소스 그룹]: 03_eventGridRg
 - [토픽 세부 정보 - 이름]: new-cars
 - [토픽 세부 정보 - 지역]: Korea Central

3. [네트워킹] 탭에서 기본 설정을 유지하고 [다음]을 클릭합니다.



토픽 만들기 ...

이벤트 표

기본 네트워킹 고급 태그 검토 + 만들기

네트워크 연결

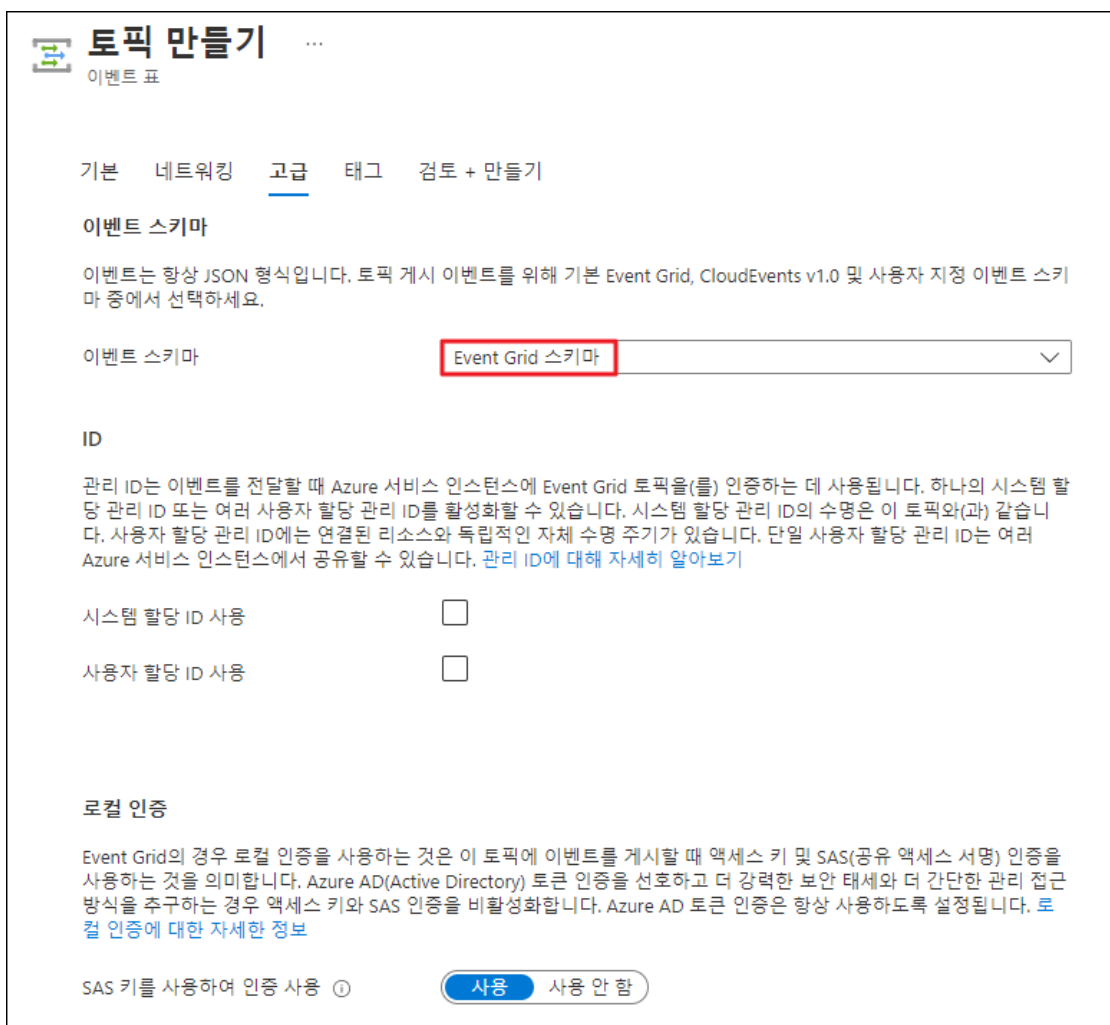
이 리소스에는 공용 IP 주소를 통해 공개적으로 또는 프라이빗 엔드포인트를 사용하여 비공개로 연결할 수 있습니다.

연결 방법

☒ 공용 액세스

☐ 프라이빗 액세스

4. [고급] 탭에서 이벤트 스키마를 기본값으로 유지하고 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.



토픽 만들기 ...

이벤트 표

기본 네트워킹 고급 태그 검토 + 만들기

이벤트 스키마

이벤트는 항상 JSON 형식입니다. 토픽 게시 이벤트를 위해 기본 Event Grid, CloudEvents v1.0 및 사용자 지정 이벤트 스키마 중에서 선택하세요.

이벤트 스키마 Event Grid 스키마 ▼

ID

관리 ID는 이벤트를 전달할 때 Azure 서비스 인스턴스에 Event Grid 토픽을(를) 인증하는 데 사용됩니다. 하나의 시스템 할당 관리 ID 또는 여러 사용자 할당 관리 ID를 활성화할 수 있습니다. 시스템 할당 관리 ID의 수명은 이 토픽과(과) 같습니다. 사용자 할당 관리 ID에는 연결된 리소스와 독립적인 자체 수명 주기가 있습니다. 단일 사용자 할당 관리 ID는 여러 Azure 서비스 인스턴스에서 공유할 수 있습니다. [관리 ID에 대해 자세히 알아보기](#)

시스템 할당 ID 사용 ☐

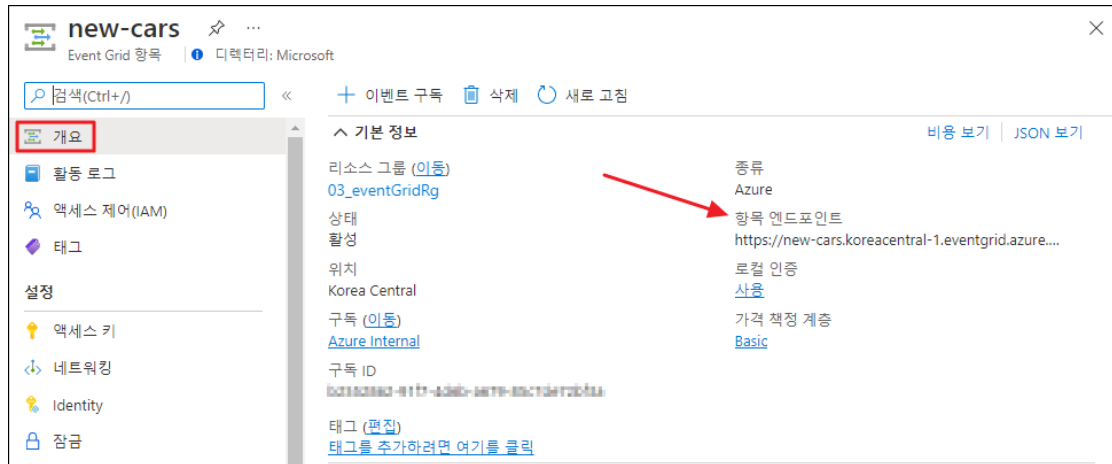
사용자 할당 ID 사용 ☐

로컬 인증

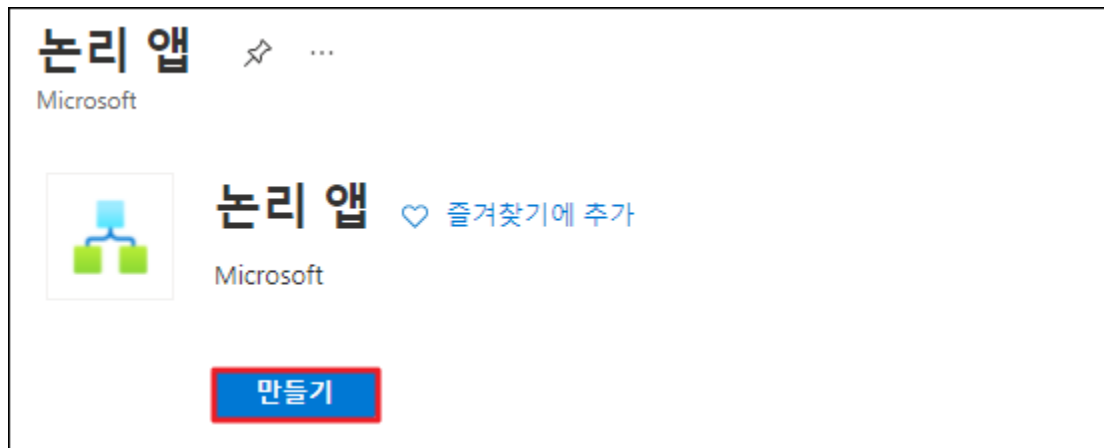
Event Grid의 경우 로컬 인증을 사용하는 것은 이 토픽에 이벤트를 게시할 때 액세스 키 및 SAS(공용 액세스 서명) 인증을 사용하는 것을 의미합니다. Azure AD(Active Directory) 토큰 인증을 선호하고 더 강력한 보안 태세와 더 간단한 관리 접근 방식을 추구하는 경우 액세스 키와 SAS 인증을 비활성화합니다. Azure AD 토큰 인증은 항상 사용하도록 설정됩니다. [로컬 인증에 대한 자세한 정보](#)

SAS 키를 사용하여 인증 사용 ① 사용 사용 안 함

5. 새로 만든 [new-cars Event Grid 항목] 블레이드로 이동합니다. 항목 엔드포인트(Topic Endpoint)에서 표시되는 엔드포인트를 확인할 수 있으며 이 엔드포인트로 이벤트를 보낼 수 있습니다.



6. 실제로 이벤트를 전송하는 새 Logic App을 만들어야 합니다. [리소스 만들기]를 클릭한 후 "논리 앱"을 검색하고 [논리 앱] 블레이드에서 [만들기]를 클릭합니다. 이 Logic App을 사용하여 Event Grid 토픽에 커스텀 이벤트를 전송할 것입니다.



7. [논리 앱 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다.
- [프로젝트 세부 정보 - 리소스 그룹]: 03_eventGridRg
 - [인스턴스 정보 - 유형]: 소비
 - [인스턴스 정보 - 논리 앱 이름]: sendEvents
 - [인스턴스 정보 - 지역]: Korea Central
 - [인스턴스 정보 - Log analytics 사용 설정]: 아니요

논리 앱 만들기 ...

기본 태그 검토 + 만들기

손쉬운 리소스 관리, 배포 및 공유를 위해 논리적 단위로 워크플로를 그룹화할 수 있는 논리 앱을 만듭니다. 워크플로를 사용하면 업무상 중요한 앱 및 서비스를 Azure Logic Apps에 연결하고 코드를 한 줄도 작성하지 않고 워크플로를 자동화할 수 있습니다.

프로젝트 세부 정보

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

구독 * ① Azure Internal

리소스 그룹 * ① 03_eventGridRg
[새로 만들기](#)

인스턴스 정보

유형 * ☒ 소비 ☐ 표준
 ⓘ 클래식 소비 경험 만들기를 찾고 계십니까? [여기를 클릭](#)

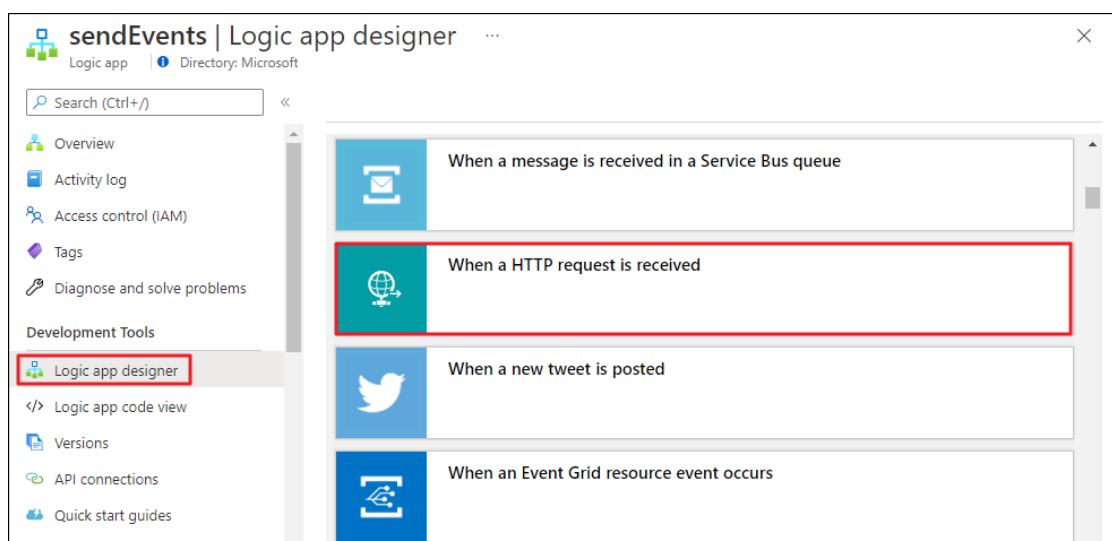
논리 앱 이름 * sendEvents ✓

지역 * Korea Central ✓

Log analytics 사용 설정 * ☐ 예 ☒ 아니요

⚠ 선택한 구독에 Log analytics 작업 영역 리소스가 없습니다. Log analytics를 사용 설정하려면 새 Log analytics 작업 영역 리소스를 만들거나 이미 있는 구독으로 전환하세요.

8. 새로 만든 [sendEvents Logic app] 블레이드의 [Development Tools - Logic app designer]로 이동합니다. 수동으로 Logic App을 실행할 것이기 때문에 [When a HTTP request is received] 트리거를 클릭합니다.



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

When a HTTP request is received

HTTP POST URL URL will be generated after save

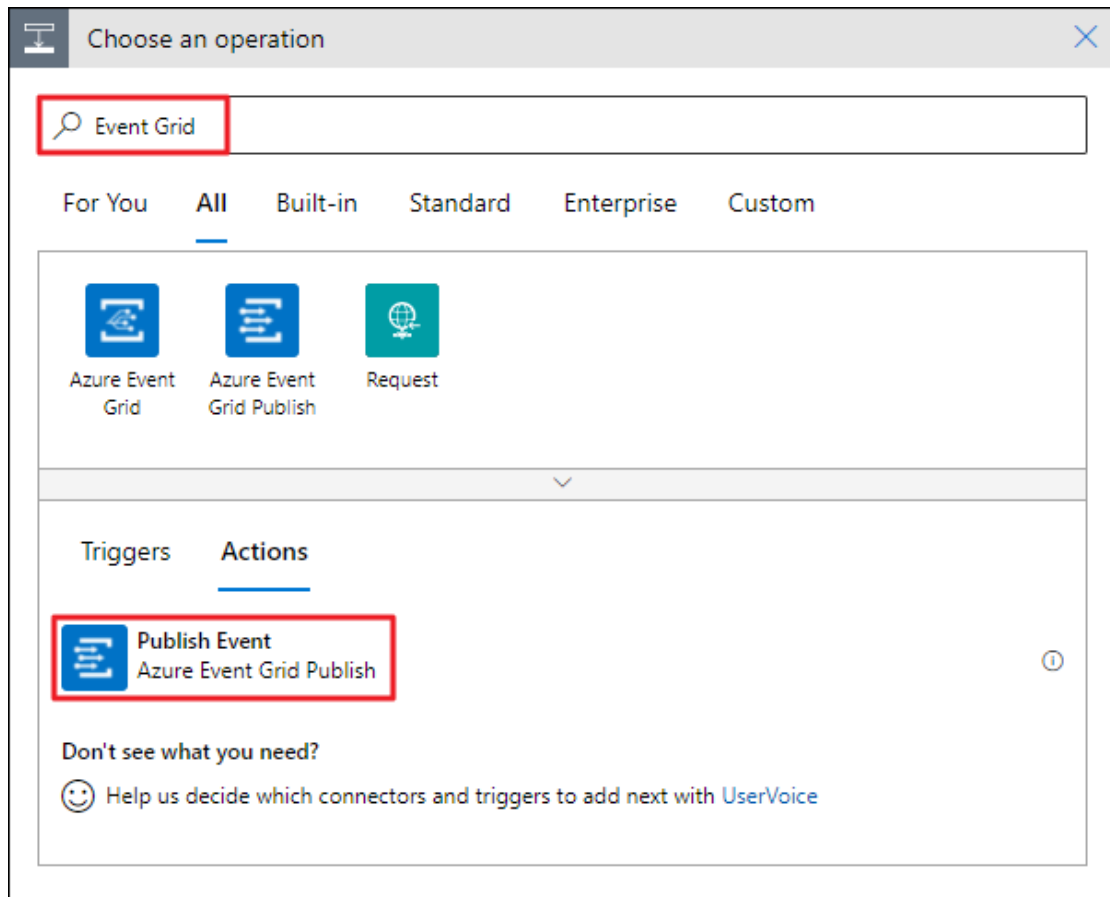
Request Body JSON Schema

Use sample payload to generate schema

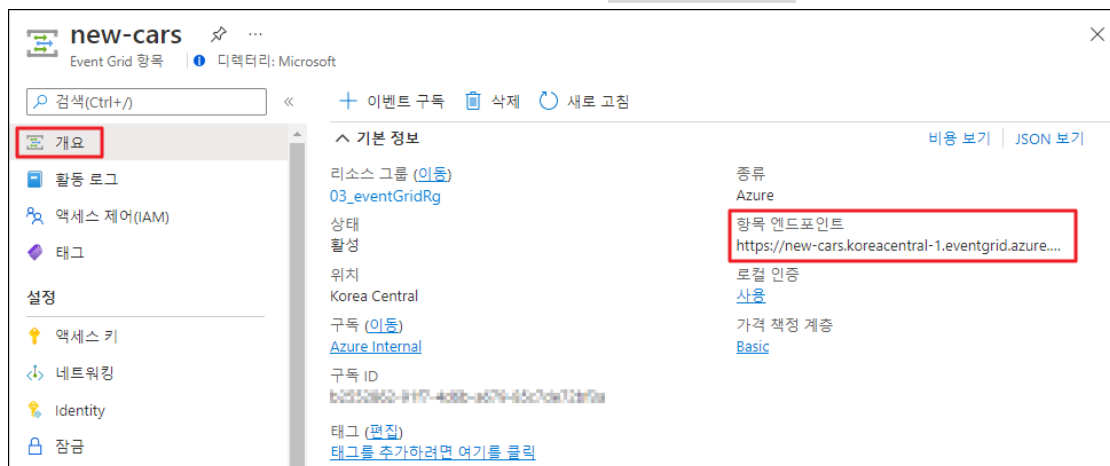
Add new parameter

+ New step

10. [Choose an operation] 타일의 검색창에서 "Event Grid"를 검색한 후 [Publish Event] 작업을 클릭합니다.



11. 브라우저에서 새 탭을 열고 Azure 포털로 이동합니다. 앞서 만들었던 [new-cars Event Grid 항목] 블레이드의 [개요]로 이동한 후 표시되는 항목 엔드포인트(Topic Endpoint) 값을 메모장에 복사합니다.



12. [new-cars Event Grid 항목] 블레이드의 [설정 - 액세스 키]로 이동한 후 “키 1”의 값을 메모장에 복사합니다.



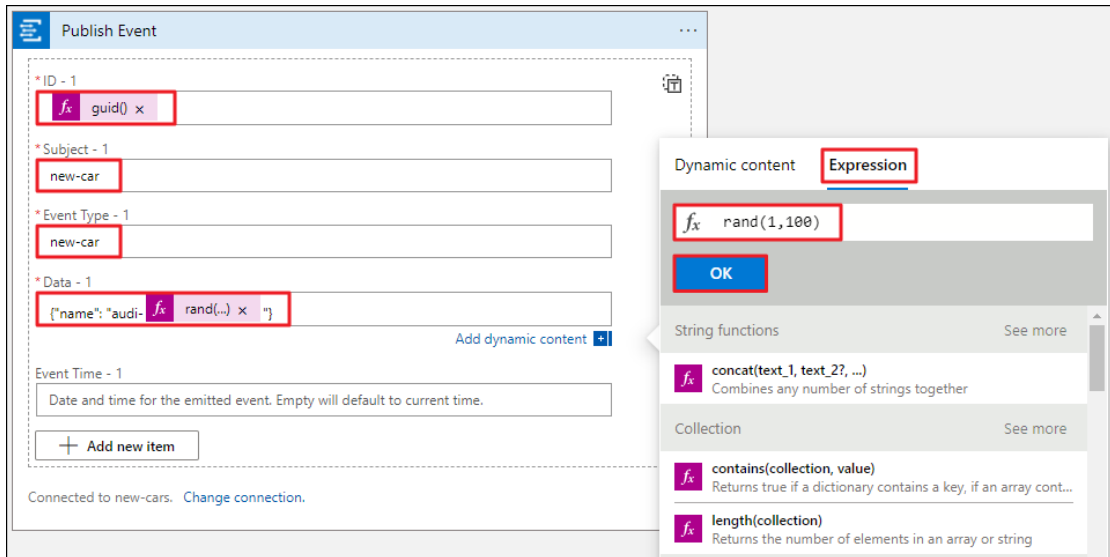
13. [sendEvents Logic app] 블레이드의 Logic App 디자이너가 실행 중인 브라우저 탭으로 이동합니다.

[Azure Event Grid Publish] 타일에서 아래와 같이 구성한 후 [Create]를 클릭합니다.

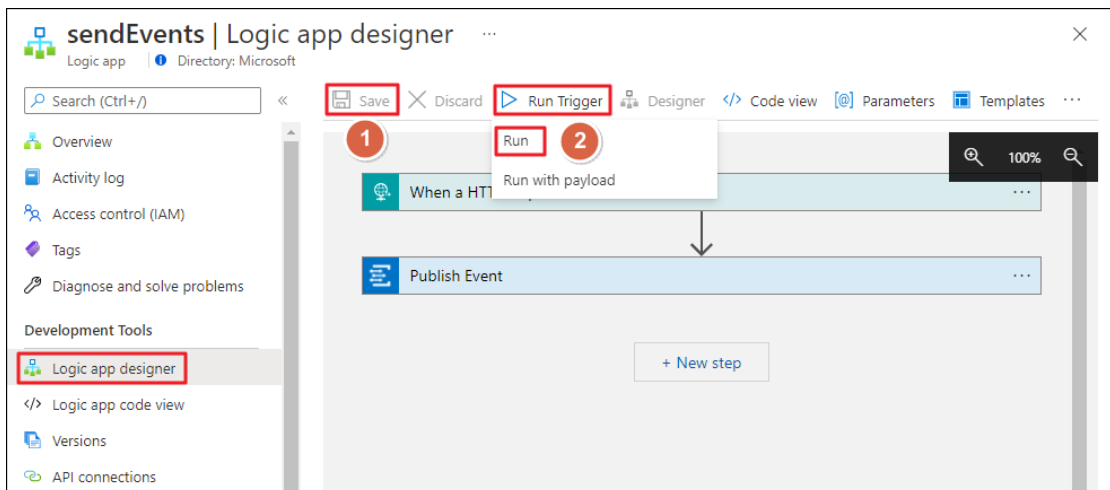
- Connection name: new-cars
- Topic Endpoint: Event Grid 항목에서 복사한 항목 엔드포인트 값을 붙여 넣습니다.
- Shared Access Signature: Event Grid 항목의 액세스 키에서 복사한 값을 붙여 넣습니다.

14. [Publish Event] 타일에서 아래와 같이 구성합니다.

- ID - 1: 텍스트 입력창을 클릭한 후 [Expression] 탭에서 "guid()"를 입력한 후 [OK]를 클릭합니다.
- Subject - 1: new-car
- Event Type - 1: new-car. 이벤트 유형은 필터링을 작성하는 경우 사용할 수 있는 값입니다.
- Data - 1: JSON 형식의 값을 입력합니다. {"name": "audi-"} 값을 입력한 후 audi- 뒤에 커서를 위치한 상태에서 [Expression] 탭을 클릭합니다. [Expression] 탭에 rand(1,100) 표현식을 입력하고 [OK]를 클릭합니다. 결과적으로 아래 이미지와 같은 값이 입력된 것을 확인합니다.



15. [sendEvents Logic app] 블레이드의 메뉴에서 [Save]를 클릭한 후 [Run Trigger - Run]을 클릭하여 Logic App을 실행합니다.



16. Logic App 실행이 완료되면 [Publish Event] 타일을 확장합니다. 아래 예제의 경우 "audi-36"이라는 이벤트를 Event Grid로 전송하였습니다.

When a HTTP request is received 0s

Publish Event 0s

INPUTS Show raw inputs >

Body

```
{
  "data": {
    "name": "audi-36"
  },
  "eventType": "new-car",
  "id": "210184bf-00d0-4de4-852e-b66a910fc4dd",
  "subject": "new-car"
}
```

OUTPUTS Show raw outputs >

Status code

200

Headers

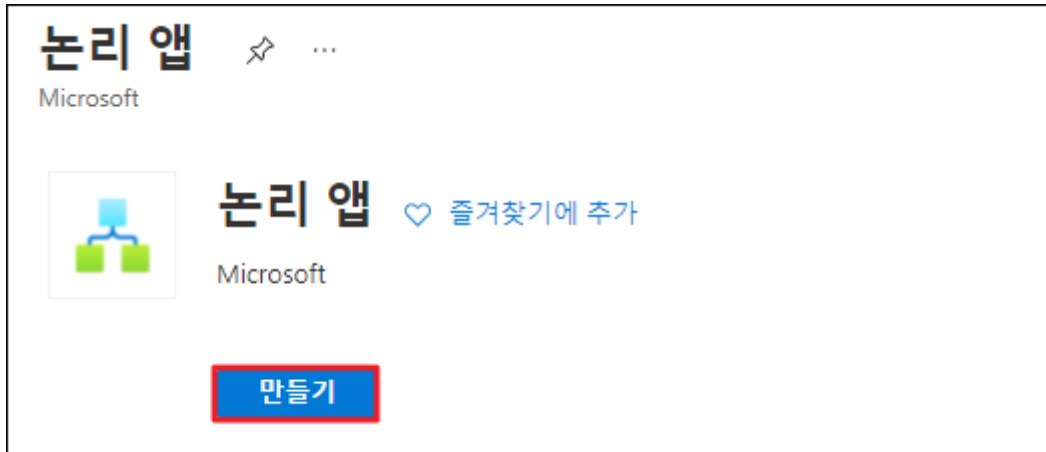
Key	Value
Strict-Transport-Security	max-age=31536000; includeS...
api-supported-versions	2018-01-01
x-ms-request-id	8207af01-d171-412d-bbcc-63...

17. 이제 이벤트를 토픽으로 보냈습니다. 하지만 어떤 구독(subscription)도 해당 이벤트를 수신하지 않기 때문에 작업이 완료된 것은 아닙니다.

TASK 03. 커스텀 이벤트(custom event)를 구독(subscribe)

이 작업에서는 앞서 만들었던 커스텀 이벤트에 대한 구독(subscription)을 추가합니다. 이 작업에서는 이벤트를 수신하는 애플리케이션인 이벤트 핸들러를 Logic App으로 설정합니다.

1. 먼저 이벤트 핸들러 역할을 할 Logic App을 하나 더 만들어야 합니다. [리소스 만들기]를 클릭한 후 "논리 앱"을 검색하고 [논리 앱] 블레이드에서 [만들기]를 클릭합니다.



2. [논리 앱 만들기] 블레이드의 [기본] 탭에서 아래와 같이 구성한 후 [검토 + 만들기]를 클릭합니다. [검토 + 만들기] 탭에서 [만들기]를 클릭합니다. 이 Logics App은 애플리케이션의 Restful 엔드포인트를 시뮬레이션하기 위해 사용합니다.
 - [프로젝트 세부 정보 - 리소스 그룹]: 03_eventGridRg
 - [인스턴스 정보 - 유형]: 소비
 - [인스턴스 정보 - 논리 앱 이름]: webhookHandler
 - [인스턴스 정보 - 지역]: Korea Central
 - [인스턴스 정보 - Log analytics 사용 설정]: 아니요

논리 앱 만들기 ...

기본 태그 검토 + 만들기

손쉬운 리소스 관리, 배포 및 공유를 위해 논리적 단위로 워크플로를 그룹화할 수 있는 논리 앱을 만듭니다. 워크플로를 사용하면 업무상 중요한 앱 및 서비스를 Azure Logic Apps에 연결하고 코드를 한 줄도 작성하지 않고 워크플로를 자동화할 수 있습니다.

프로젝트 세부 정보

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

구독 * ① Azure Internal ▼

리소스 그룹 * ① 03_eventGridRg ▼
[새로 만들기](#)

인스턴스 정보

유형 * ☒ 소비 ☐ 표준
 ⓘ 클래식 소비 경험 만들기를 찾고 계십니까? [여기를 클릭](#)

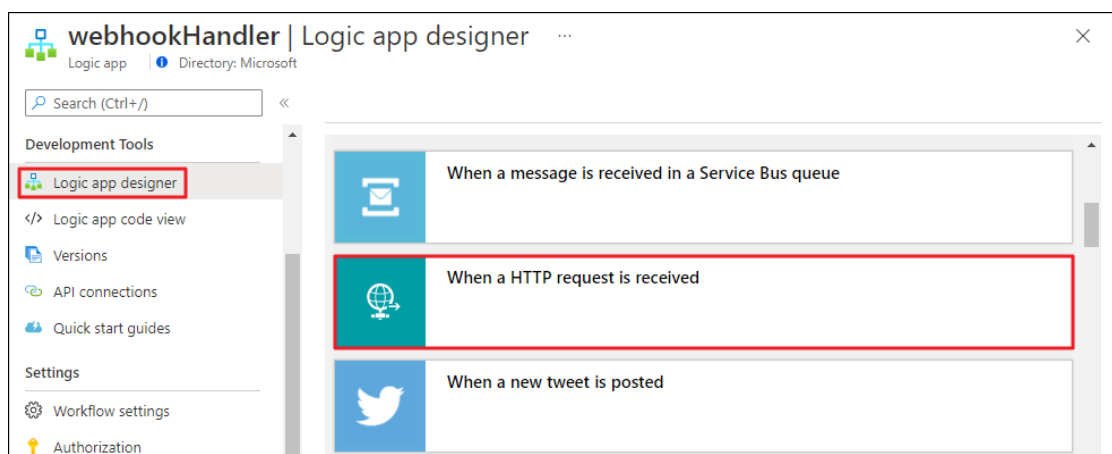
논리 앱 이름 * webhookHandler ✓

지역 * Korea Central ▼

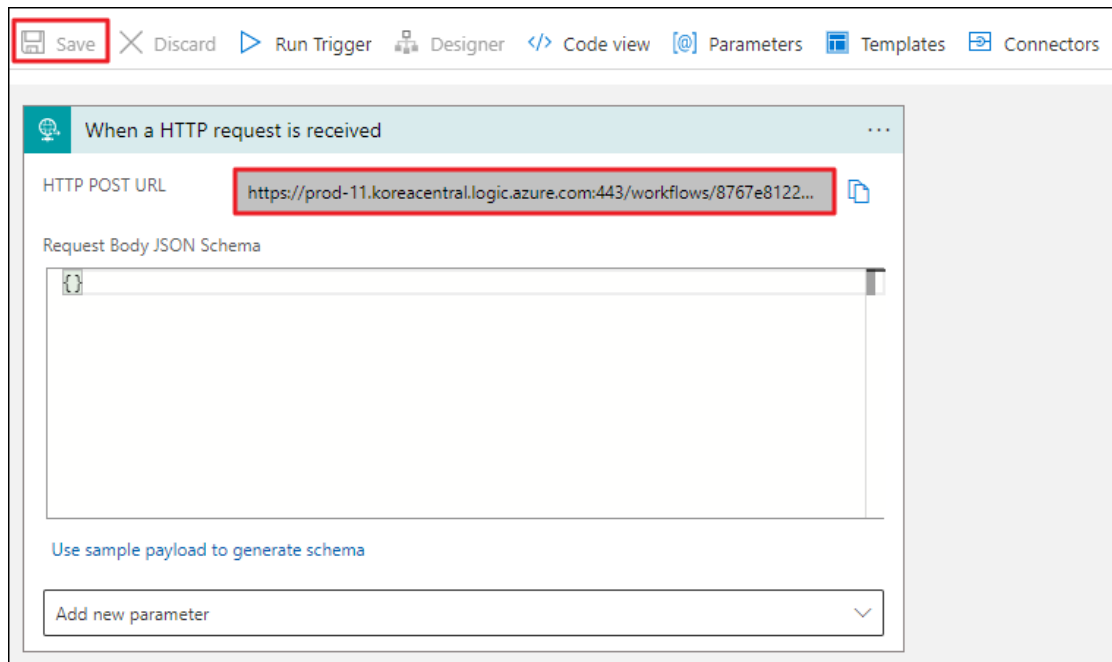
Log analytics 사용 설정 * ☐ 예 ☒ 아니요

⚠ 선택한 구독에 Log analytics 작업 영역 리소스가 없습니다. Log analytics를 사용 설정하려면 새 Log analytics 작업 영역 리소스를 만들거나 이미 있는 구독으로 전환하세요.

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



4. [When a HTTP request is received] 타일에서 아무런 설정도 하지 않고 Logic App 디자이너의 메뉴에서 [Save]를 클릭합니다. [When a HTTP request is received] 타일에서 Logic App에 대한 고유한 webhook URL이 생성된 것을 확인합니다. 이 webhook URL을 클립보드에 복사합니다.

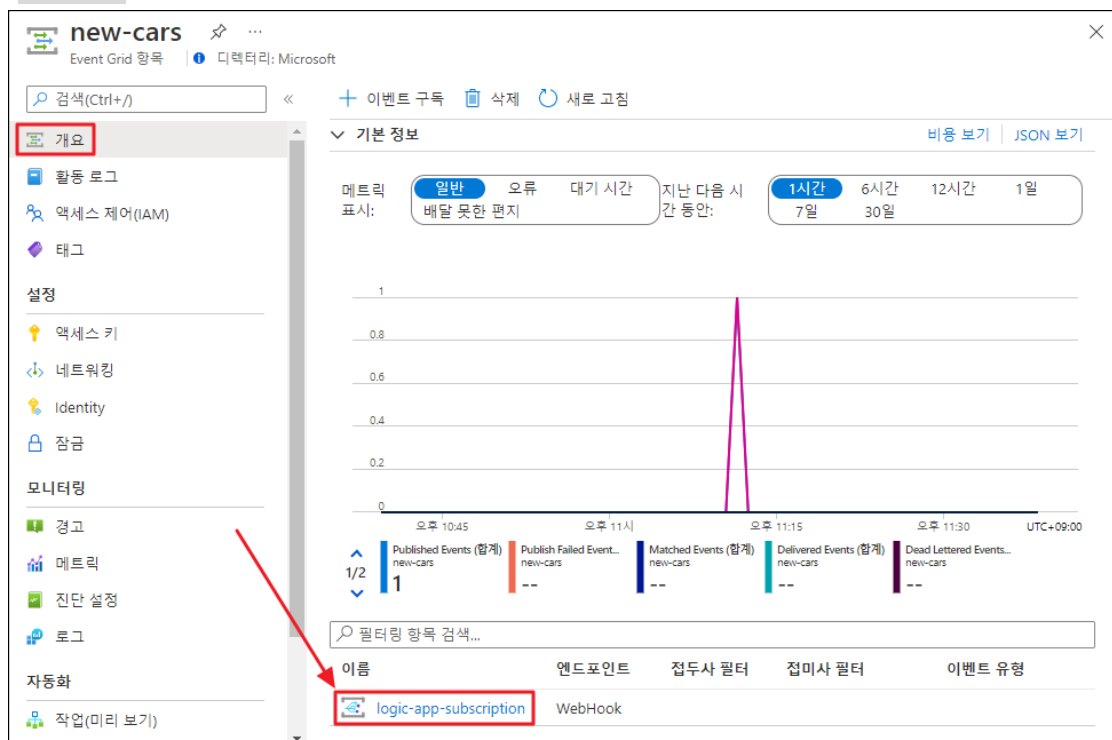


5. [new-cars Event Grid 항목] 블레이드의 [개요]로 이동한 후 메뉴에서 [이벤트 구독]을 클릭합니다.

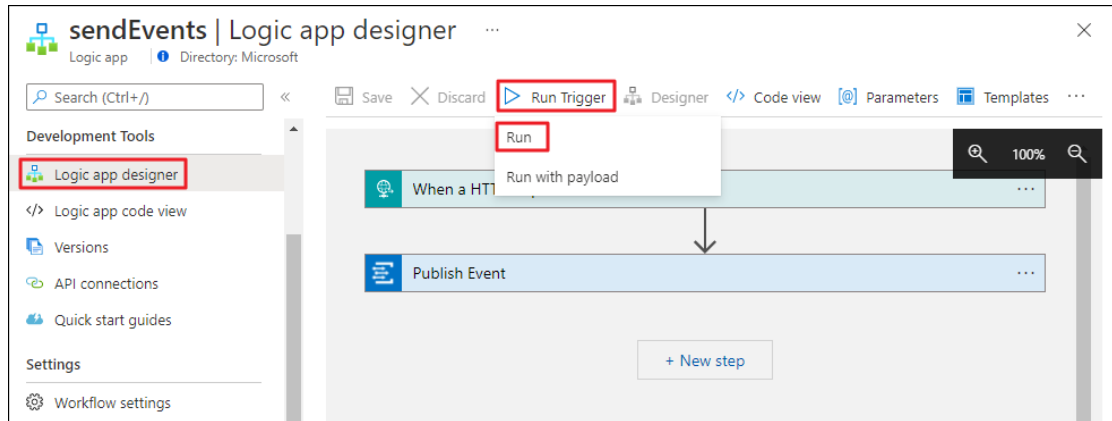


6. [이벤트 구독 만들기] 블레이드에서 아래와 같이 구성한 후 [필터], [추가 기능], [전송 속성], [고급 편집기] 탭의 내용을 추가로 검토하고 [만들기]를 클릭합니다.
- [이벤트 구독 정보 - 이름]: logic-app-subscription
 - [이벤트 구독 정보 - 이벤트 스키마]: Event Grid 스키마
 - [엔드포인트 정보 - 엔드포인트 유형]: "웹 후크"를 선택합니다. Azure Function, Storage Queue, Event Hubs, 하이브리드 연결, Service Bus Queue, Service Bus Topic 등 다양한 엔드포인트 유형을 사용할 수 있습니다.
 - [엔드포인트 정보 - 엔드포인트]: "엔드포인트 선택" 링크를 클릭합니다. [웹 후크 선택]에서 앞서 Logic App에서 복사했던 webhook URL 값을 붙여 넣고 [선택 확인]을 클릭합니다.

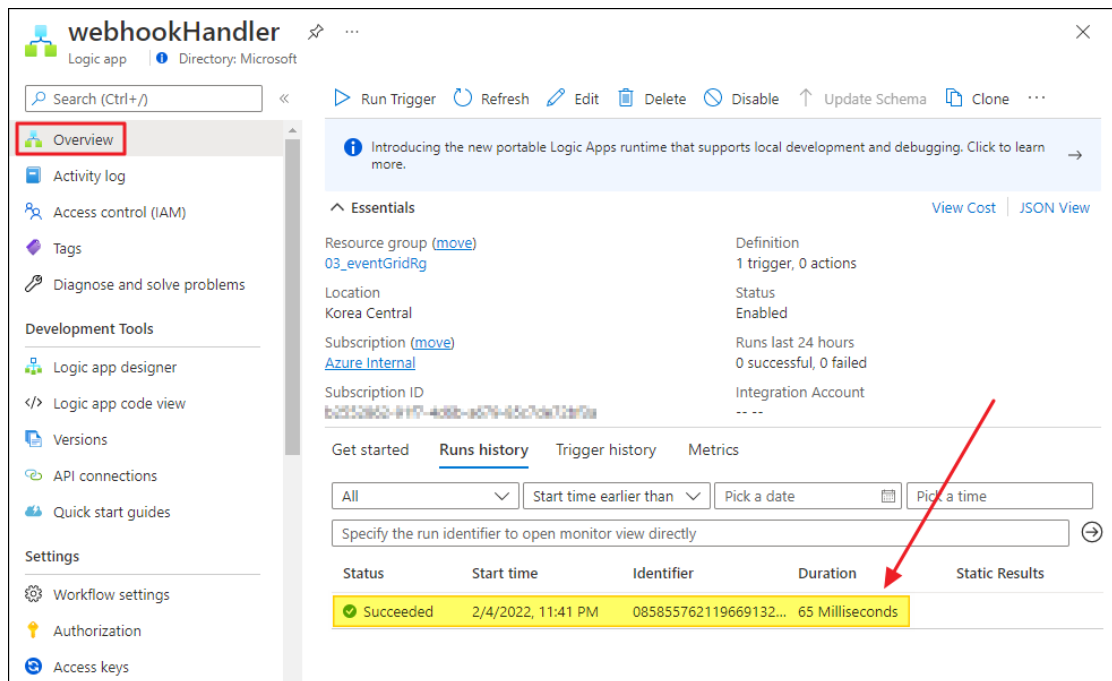
7. [new-cars Event Grid 항목] 블레이드의 [개요]에서 아래와 같이 새로 추가된 구독을 확인할 수 있습니다.



8. 이제 이벤트를 전송하는 Logic App에서 다시 한번 더 이벤트를 전송합니다. [sendEvents Logic app] 블레이드의 [Development Tools - Logic app designer]로 이동한 후 메뉴에서 [Run Trigger - Run]을 클릭합니다.



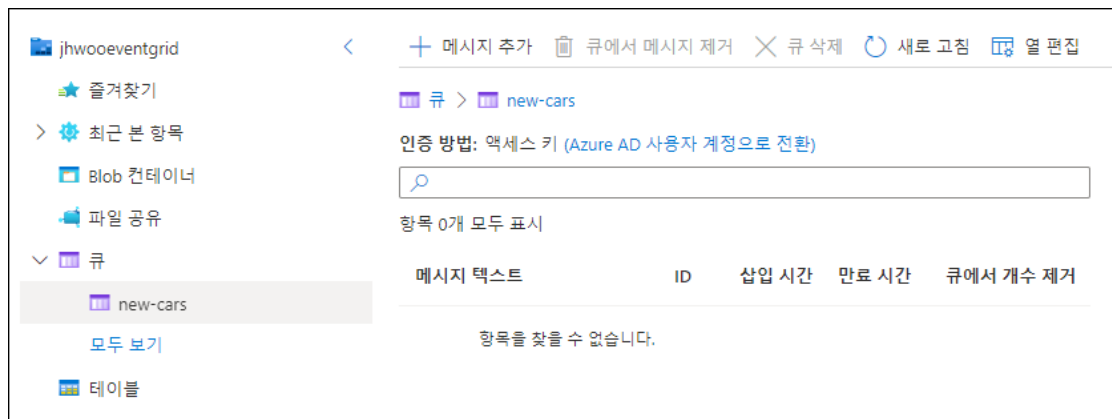
9. 이벤트를 수신하는 이벤트 핸들러 Logic App에서 정상적으로 이벤트를 수신하였는지 확인해야 합니다. [webhookHandler Logic app] 블레이드의 [Overview]로 이동한 후 메뉴에서 [Refresh]를 클릭합니다. 아래와 같이 새로운 이벤트가 수신된 것을 확인할 수 있습니다.



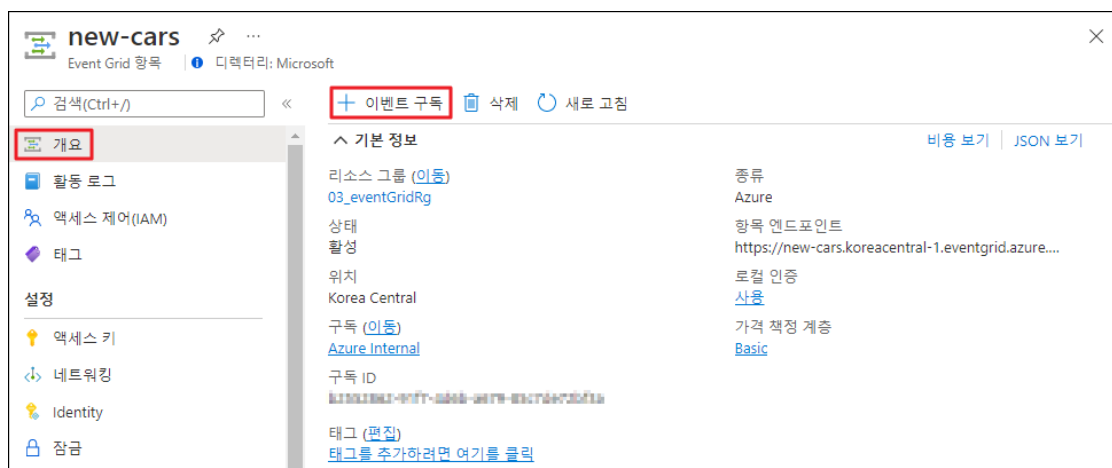
10. Event Grid에 여러 구독자(subscriber)를 추가하는 것은 매우 간단합니다. 이 작업에서는 앞서 만들었던 스토리지 계정의 큐를 새 구독자로 추가합니다. [스토리지 계정] 블레이드로 이동한 후 [스토리지 브라우저]를 클릭합니다. 스토리지 브라우저의 네비게이션에서 "큐"를 선택하고 메뉴에서 [큐 추가]를 클릭합니다. [큐 추가]에서 "new-cars"를 입력한 후 [확인]을 클릭합니다.



11. 새로 만든 "new-cars" 큐를 클릭한 후 큐에 아무런 내용도 없는 것을 확인합니다.



12. [new-cars Event Grid 항목] 블레이드로 이동한 후 메뉴에서 [이벤트 구독]을 클릭합니다. 단일 토픽에 대해 여러 구독을 가질 수 있습니다. 이를 fan-out 시나리오라고 합니다.



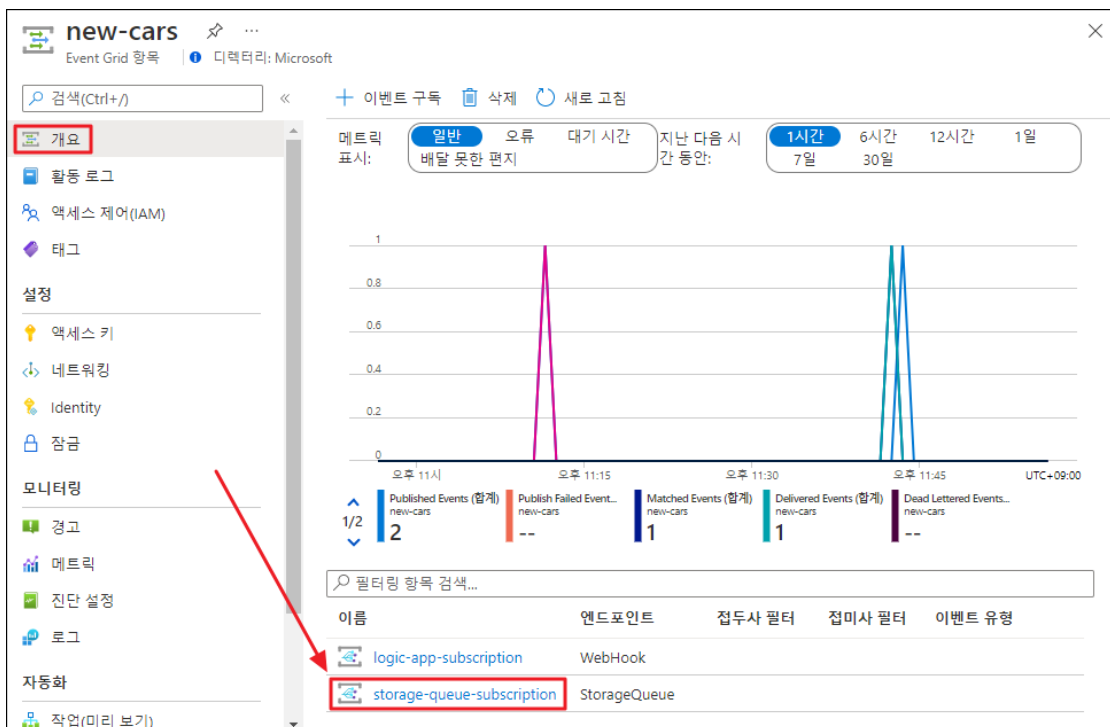
13. [이벤트 구독 만들기] 블레이드에서 아래와 같이 구성하고 [만들기]를 클릭합니다.

- [이벤트 구독 정보 - 이름]: storage-queue-subscription
- [이벤트 구독 정보 - 이벤트 스키마]: Event Grid 스키마
- [엔드포인트 정보 - 엔드포인트 유형]: 스토리지 큐
- [엔드포인트 정보 - 엔드포인트]: "엔드포인트 선택" 링크를 클릭합니다. [큐]에서 앞서 스토리지 큐를

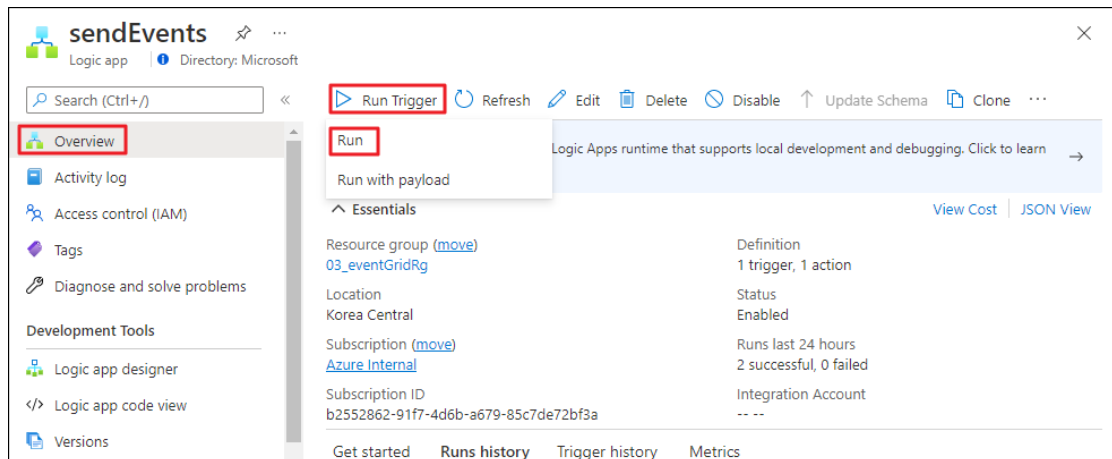
만든 스토리지 계정과 큐를 선택하고 [선택]을 클릭합니다.

- [전송용 관리 ID - 관리 ID 유형]: 없음

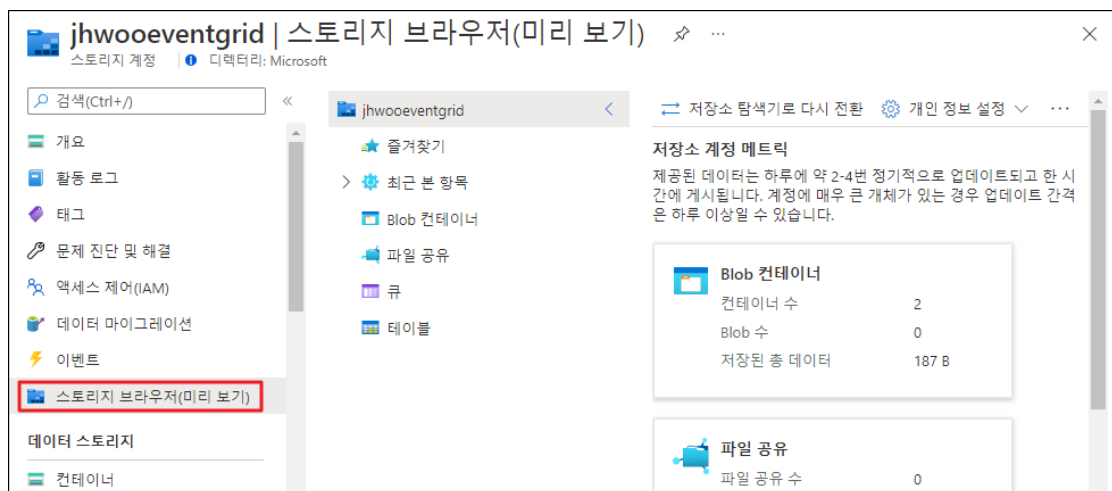
14. [new-cars Event Grid 항목] 블레이드의 [개요]에서 이제 두 개의 구독자(subscriber)가 표시되는 것을 확인합니다.



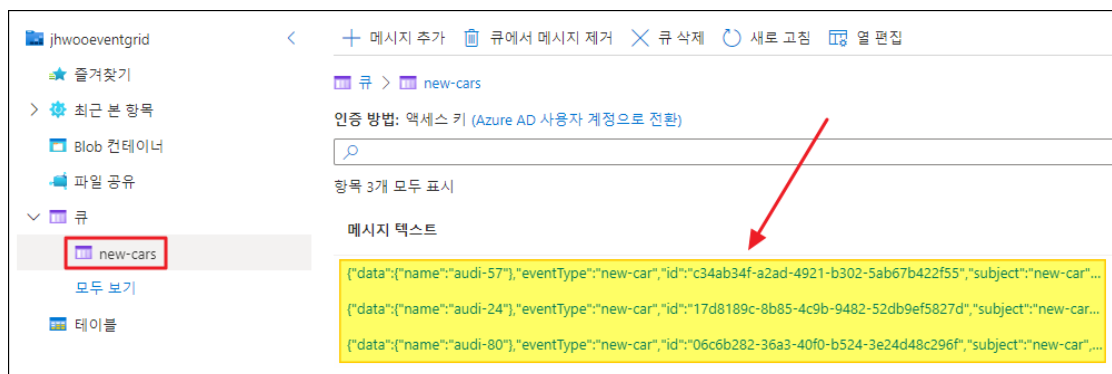
15. [sendEvents Logic app] 블레이드의 [Overview]로 이동한 후 메뉴에서 [Run Trigger - Run]을 3번 클릭합니다.



16. [스토리지 계정] 블레이드로 이동한 후 [스토리지 브라우저]를 클릭합니다.



17. 스토리지 브라우저의 [큐 - new-cars]로 이동하면 아래와 같이 큐에 3개의 이벤트가 수신된 것을 확인할 수 있습니다.



18. [webhookHandler Logic app] 블레이드의 [Overview]로 이동한 후 마찬가지로 3개의 이벤트가 수신된 것을 확인합니다. 이벤트를 클릭하면 Storage Queue와 동일한 이벤트가 수신된 것을 확인할 수 있습니다.

webhookHandler
Logic app | Directory: Microsoft

Search (Ctrl+J) << Run Trigger Refresh Edit Delete Disable Update Schema Clone >>

Introducing the new portable Logic Apps runtime that supports local development and debugging. Click to learn more. →

Essentials View Cost JSON View

Get started **Runs history** Trigger history Metrics

All Start time earlier than Pick a date Pick a time

Specify the run identifier to open monitor view directly →

Status	Start time	Identifier	Duration	Static Results
✓ Succeeded	2/4/2022, 11:58 PM	085855762017551740...	122 Milliseconds	
✓ Succeeded	2/4/2022, 11:58 PM	085855762017839970...	70 Milliseconds	
✓ Succeeded	2/4/2022, 11:58 PM	085855762018358544...	111 Milliseconds	
✓ Succeeded	2/4/2022, 11:41 PM	085855762119669132...	65 Milliseconds	