## Учреждение образования

## Белорусский государственный технологический университет

Кафедра информационных систем и технологий

**ЛАБОРАТОРНАЯ РАБОТА № 9**

по дисциплине «Корпоративные информационные системы»

Тема

«Azure разработка решений на основе событий»

Выполнил студент

4 курса 5 группы, Демьянов В.Р.

Минск 2023

**Лабораторная работа №9**

Azure разработка решений на основе событий

**Цель работы:** изучение возможностей Azure для разработки решений на основе событий.

**Ход работы**

В Azure для разработки и интеграции приложений использующие события существует высокомасштабируемый бессерверный брокер событий. События доставляются сеткой событий в пункты назначения подписчиков, такие как приложения, службы Azure или любую конечную точку, к которой у сетки событий есть сетевой доступ. Источником этих событий могут быть другие приложения, службы SaaS и службы Azure. На рисунке 1 представлена архитектура решения на основе событий.

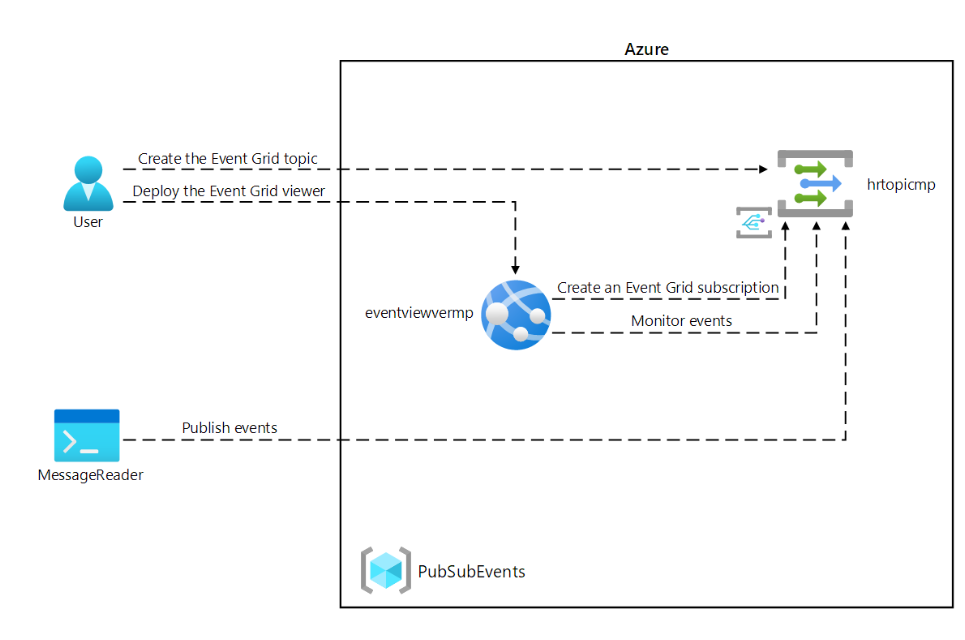


Рисунок 1 – Архитектура решения на основе событий

Перед тем как приступить к созданию такого решения, проверим зарегистрирован ли Microsoft.EventGrid провайдер ресурсов. Выполнение команды для такой проверки представлена на рисунке 2.

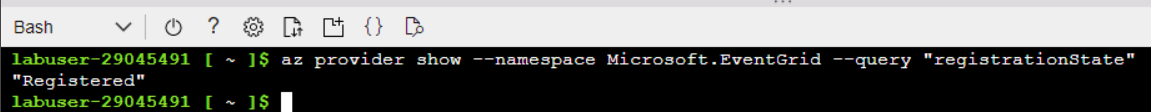


Рисунок 2 – Проверка регистрации провайдера ресурсов Microsoft.EventGrid

Далее создадим пользовательский раздел сетки событий (рисунок 3).

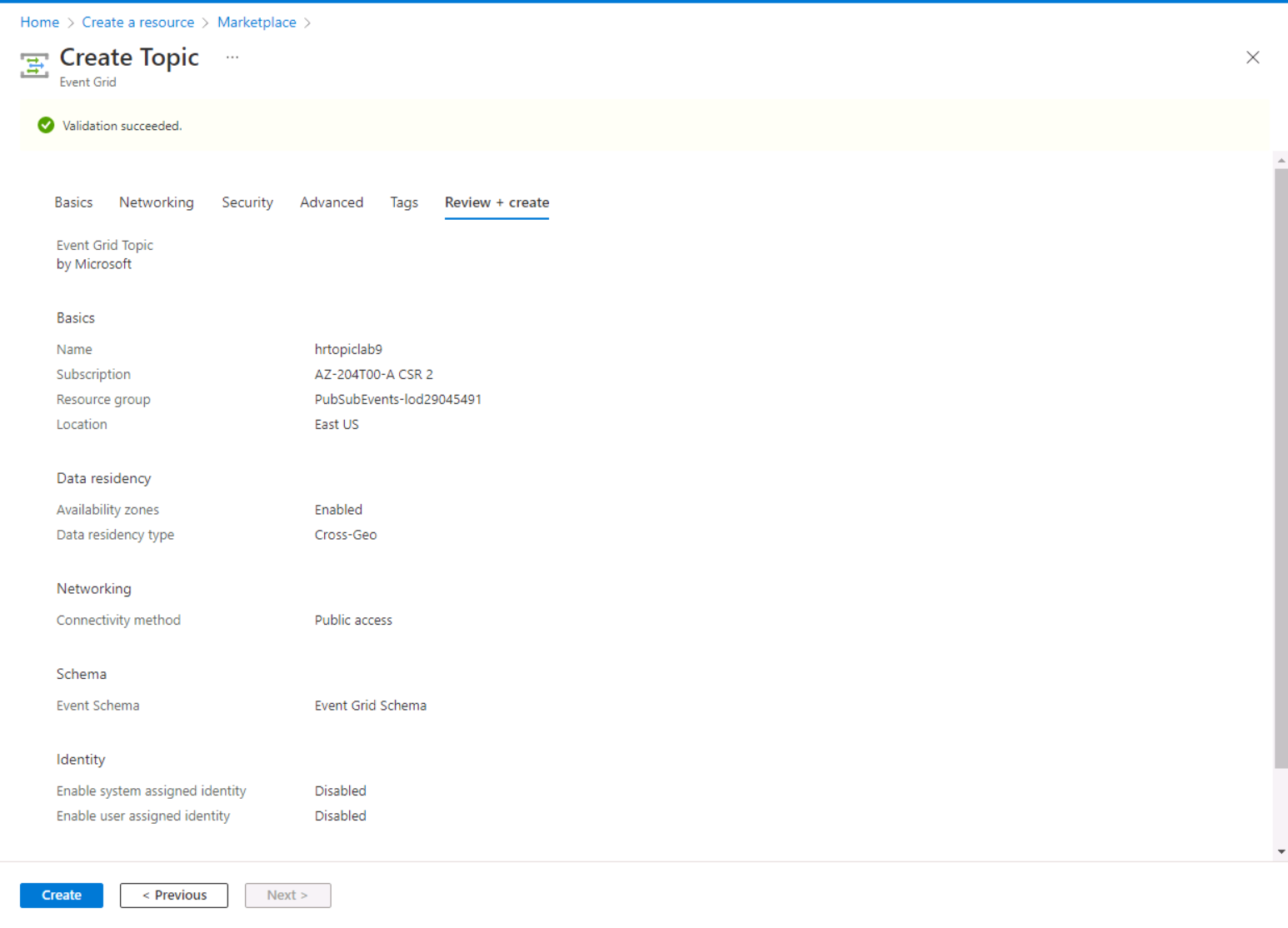


Рисунок 3 – Создание пользовательского раздела сетки событий

Раздел сетки событий предоставляет конечную точку, в которую источник отправляет события. Издатель создает раздел сетки событий и решает, сколько разделов нужно источнику событий: один или несколько. Раздел используется для коллекции связанных событий. Чтобы реагировать на события определенных типов, подписчикам следует решить, на какие из разделов подписаться.

Системные разделы – это встроенные разделы, предоставляемые службами Azure. Они не отображаются в подписке Azure, так как принадлежат издателю, но вы можете на них подписаться. Чтобы оформить подписку, укажите сведения о ресурсе, из которого вы хотите получать события. Пока у вас есть доступ к ресурсу, вы можете подписаться на его события.

Пользовательские разделы – это разделы приложений и сторонних поставщиков. После того как вы создадите или назначите доступ к пользовательскому разделу, он будет отображаться в вашей подписке.

На рисунке 4 представлен dashboard созданного раздела сетки событий.

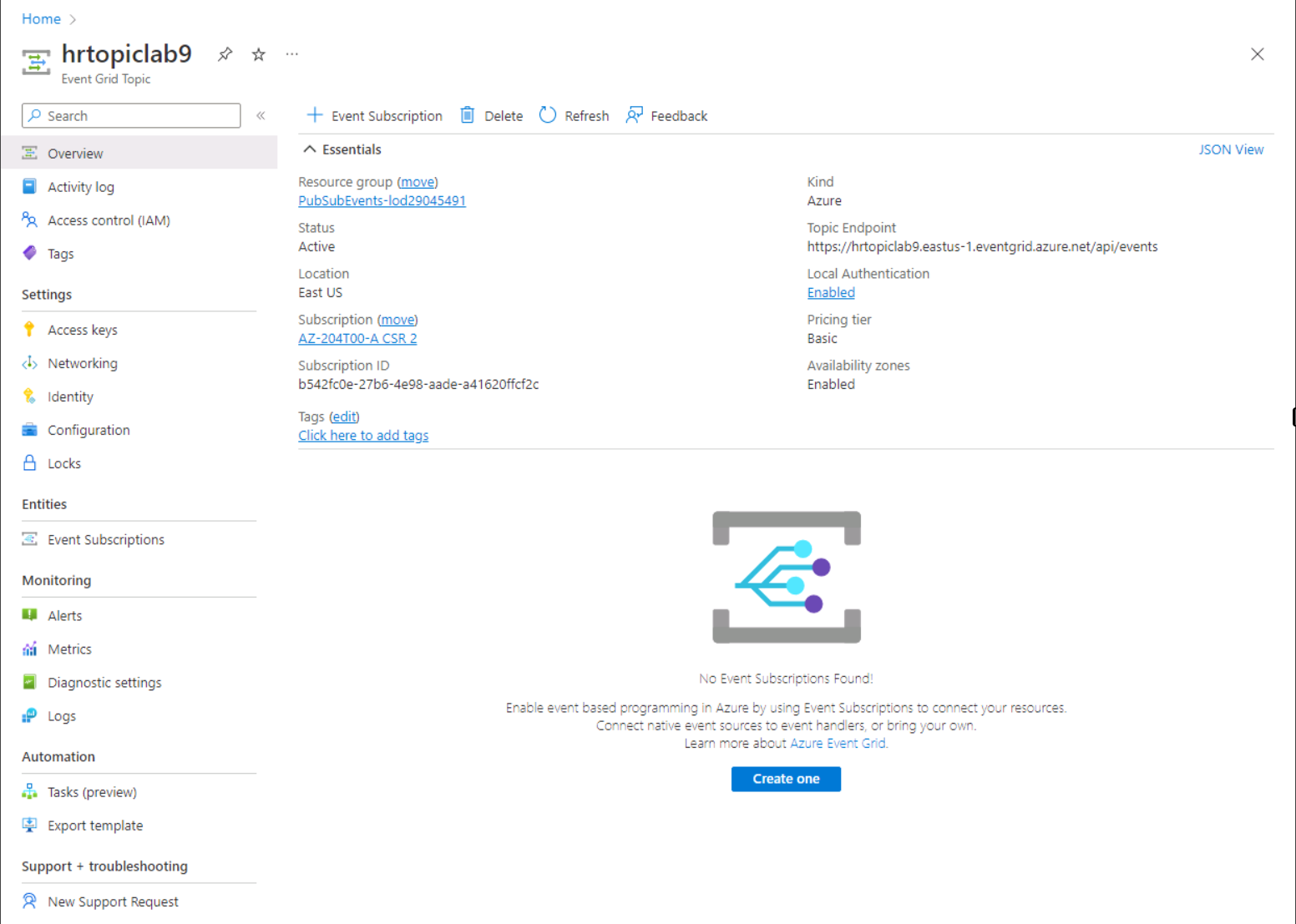


Рисунок 4 – Dashboard сетки событий

Далее создадим Web App (рисунок 5).

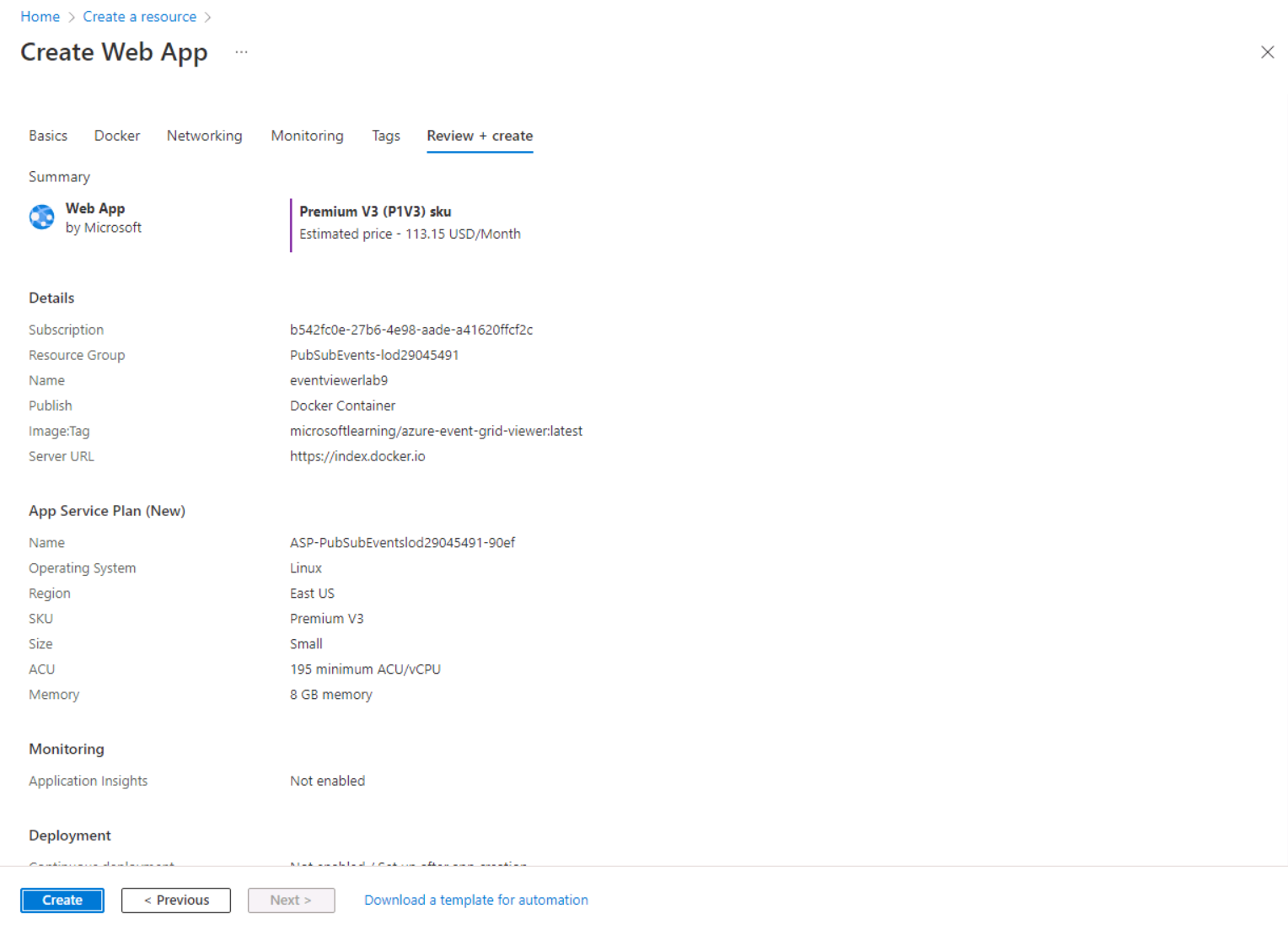


Рисунок 5 – Создание Web App

Dashboard созданного Web App представлен на рисунке 6.

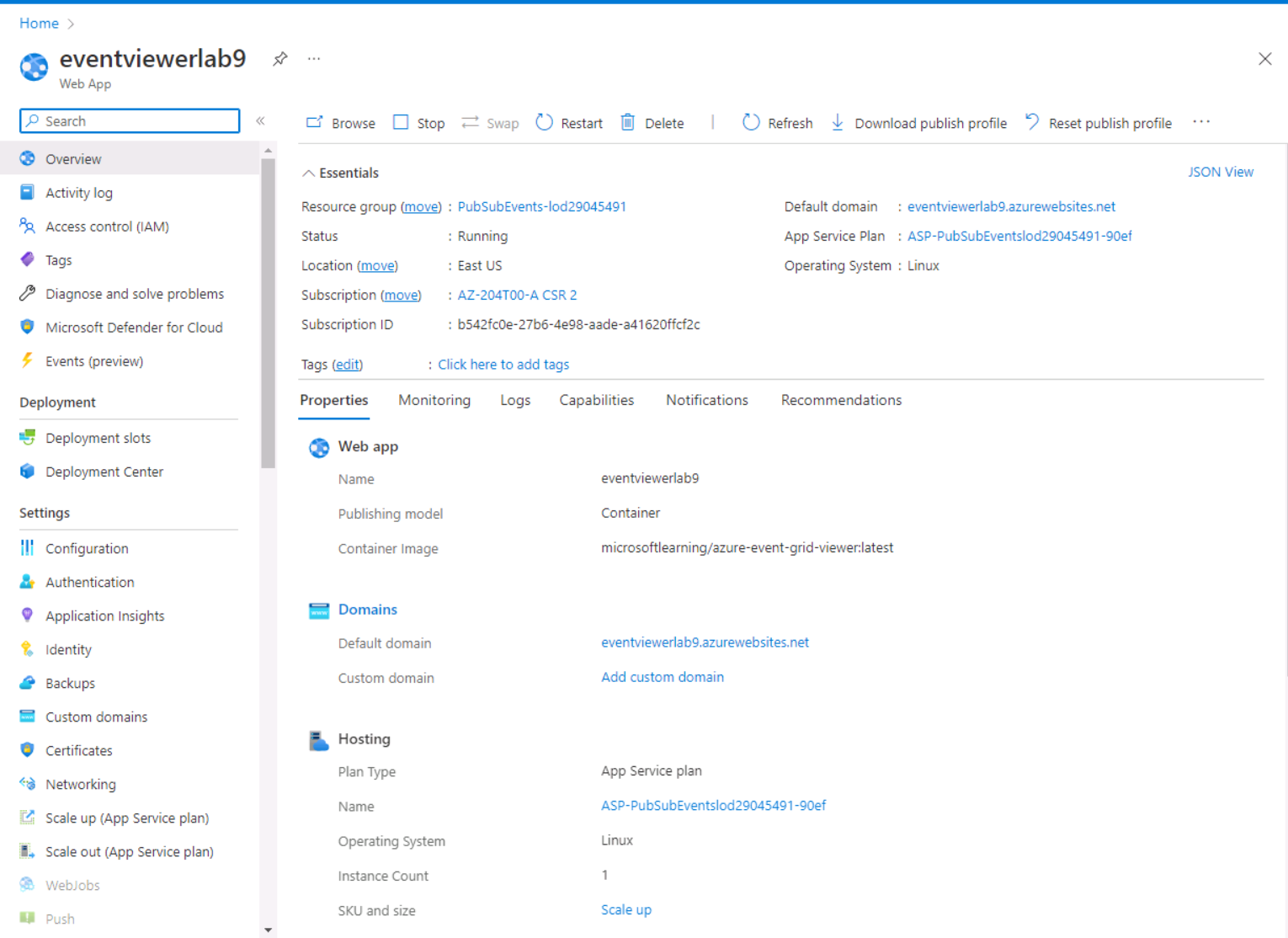


Рисунок 6 – Dashboard созданного Web App

Перед созданием подписки на Event Grid, проверим развёрнутый Azure Event Grid Viewer (рисунок 7).

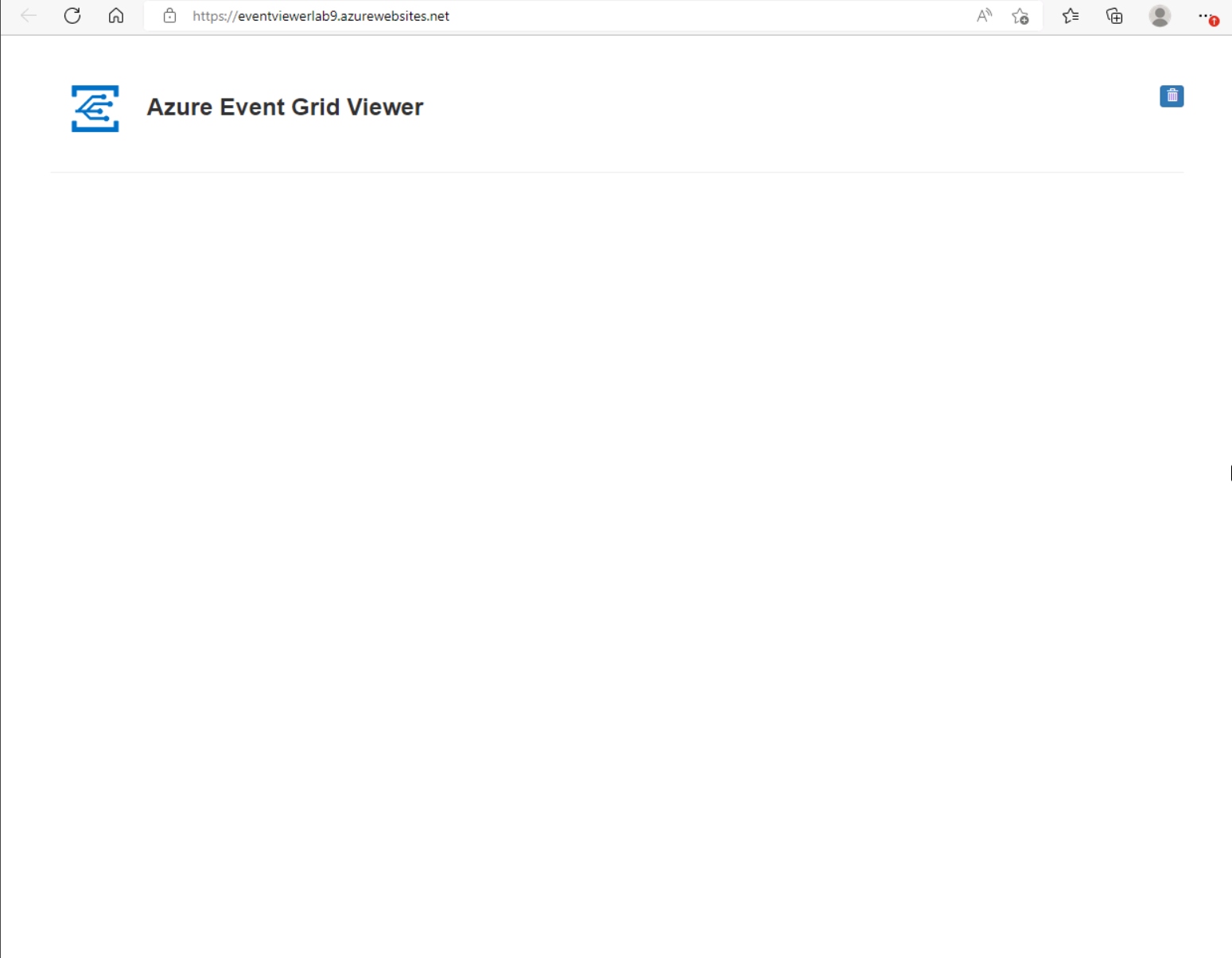


Рисунок 7 – Event Grid Viewer

Сейчас список подписок пуст. Перейдём к созданию (регистрации) новой подписки (рисунок 8).

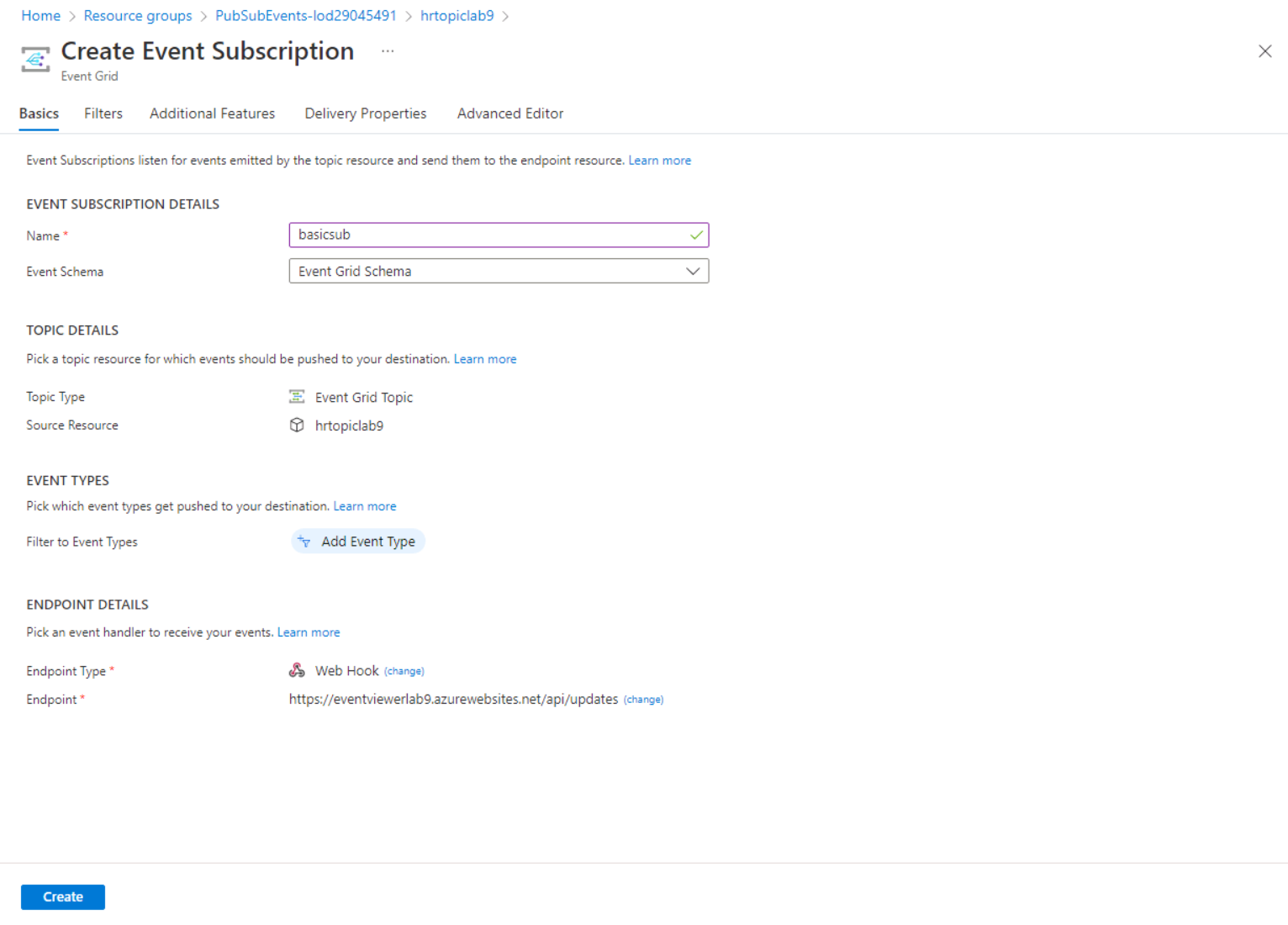


Рисунок 8 – Создание новой подписки

Подписки на события – это конечная точка или встроенный механизм для маршрутизации событий, иногда в несколько обработчиков. Кроме того, подписки используются обработчиками для интеллектуальной фильтрации входящих событий. Подписка сообщает службе "Сетка событий", какие события в разделе нужно получать. При создании подписки необходимо ввести конечную точку для обработки событий. Вы можете фильтровать события, которые отправляются в конечную точку. Можно задать фильтр по типу события или шаблону темы. Задайте срок действия подписки на событие, которая необходима только в течение ограниченного времени и не нужно беспокоиться об очистке этих подписок.

Вернёмся в Event Grid Viewer и убедимся в том, что событие зарегистрировано (рисунок 9).

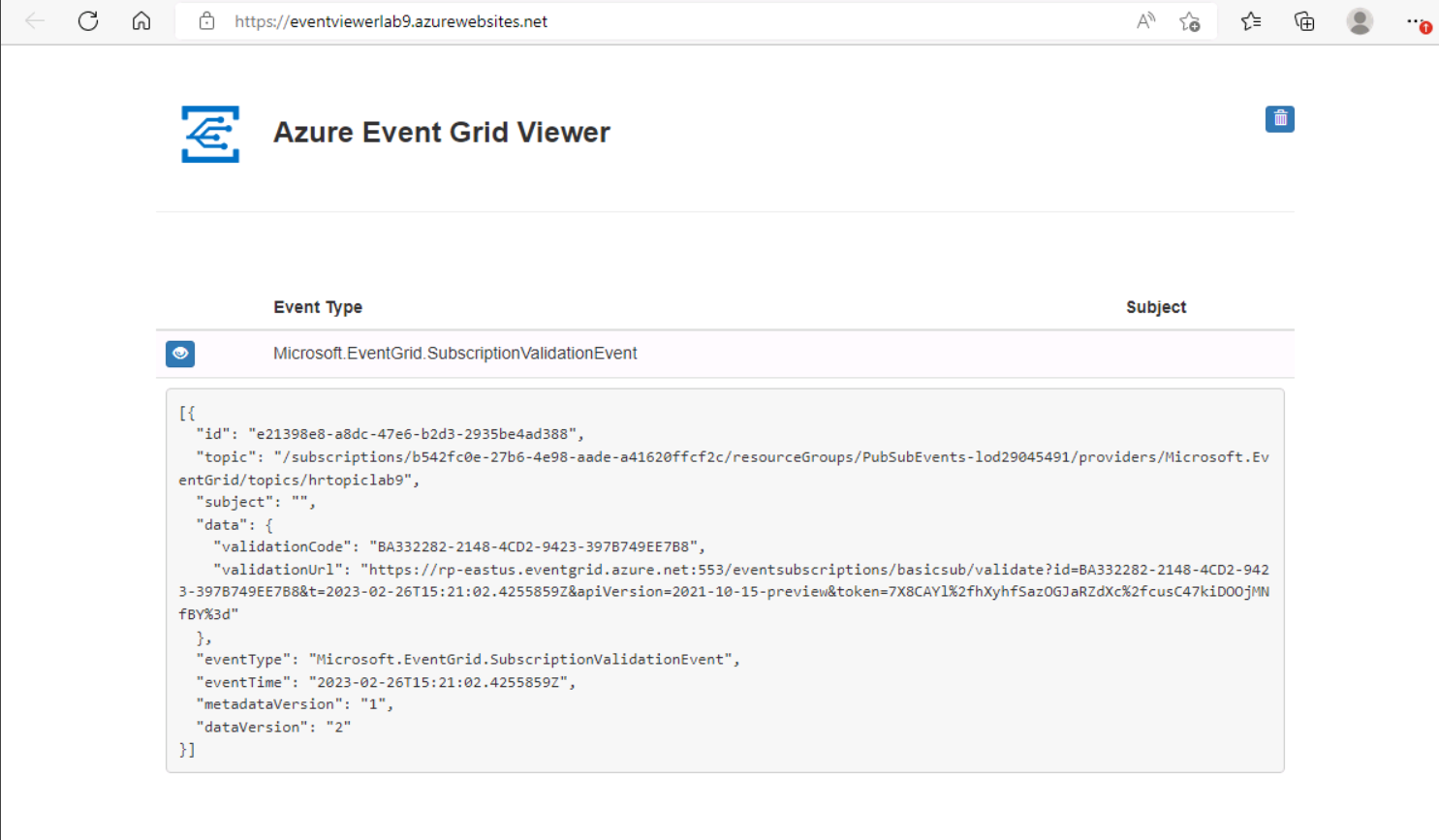


Рисунок 9 – Созданное событие

Далее записываем креды созданной подписки, так как они будут использоваться далее (рисунок 10 и 11).

#### 

Рисунок 10 – Topic Endpoint

#### 

Рисунок 11 – Topic Access keys

Перейдём к публикацим Event Grid events из .NET. Для этого нам понадобиться создать .NET проект (рисунок 12).

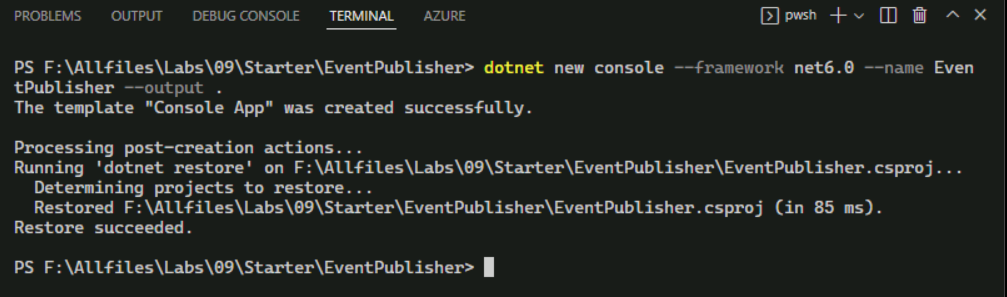


Рисунок 12 – Создание нового .NET проекта

В листинге ниже представлен код программы, которая подключается к Event Grid и публикует новые события.

|  |
| --- |
| using Azure;  using Azure.Messaging.EventGrid;  using System;  using System.Threading.Tasks;  public class Program  {      private const string topicEndpoint = "https://hrtopiclab9.eastus-1.eventgrid.azure.net/api/events";      private const string topicKey = "ArMNoNkPazLxEXWfAGf//71JhUcw40+ro0954Xgcn10=";      public static async Task Main(string[] args)      {          var endpoint = new Uri(topicEndpoint);          var credential = new AzureKeyCredential(topicKey);          var client = new EventGridPublisherClient(endpoint, credential);          var firstEvent = new EventGridEvent(              subject: $"New Employee: Alba Sutton",              eventType: "Employees.Registration.New",              dataVersion: "1.0",              data: new              {                  FullName = "Alba Sutton",                  Address = "4567 Pine Avenue, Edison, WA 97202"              }          );          var secondEvent = new EventGridEvent(              subject: $"New Employee: Alexandre Doyon",              eventType: "Employees.Registration.New",              dataVersion: "1.0",              data: new              {                  FullName = "Alexandre Doyon",                  Address = "456 College Street, Bow, WA 98107"              }          );          await client.SendEventAsync(firstEvent);          Console.WriteLine("First event published");          await client.SendEventAsync(secondEvent);           Console.WriteLine("Second event published");      }  } |

Запустим созданное приложение (рисунок 13).



Рисунок 13 – Запуск созданного приложения

Вернёмся в Event Grid Viewer и убедимся, что события опубликованы (рисунок 14).

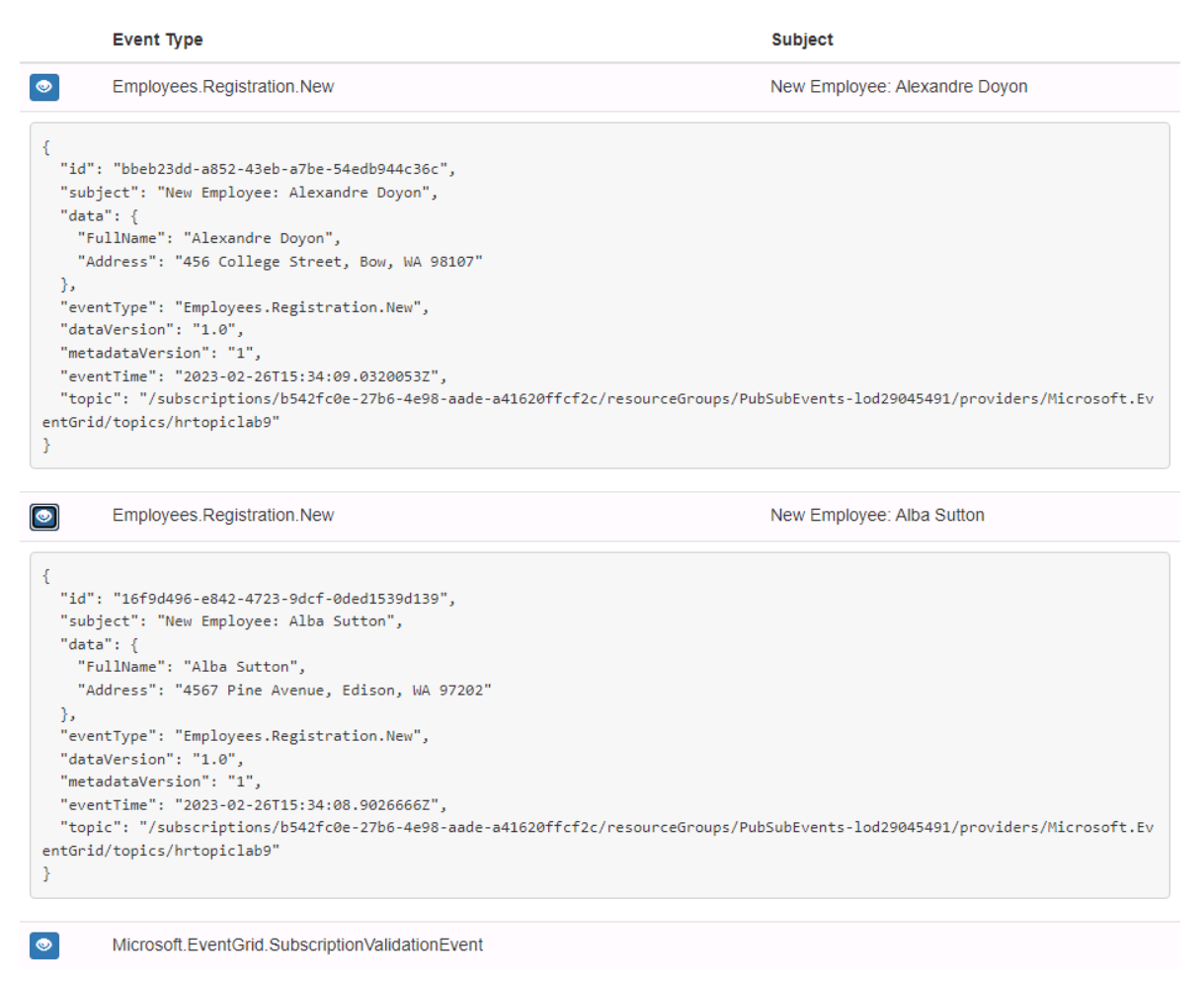


Рисунок 14 – Опубликованные события

**Вывод:** в результате выполнения лабораторной работы я ознакомился с Azure Event Grid и создал простое решение на основе событий.