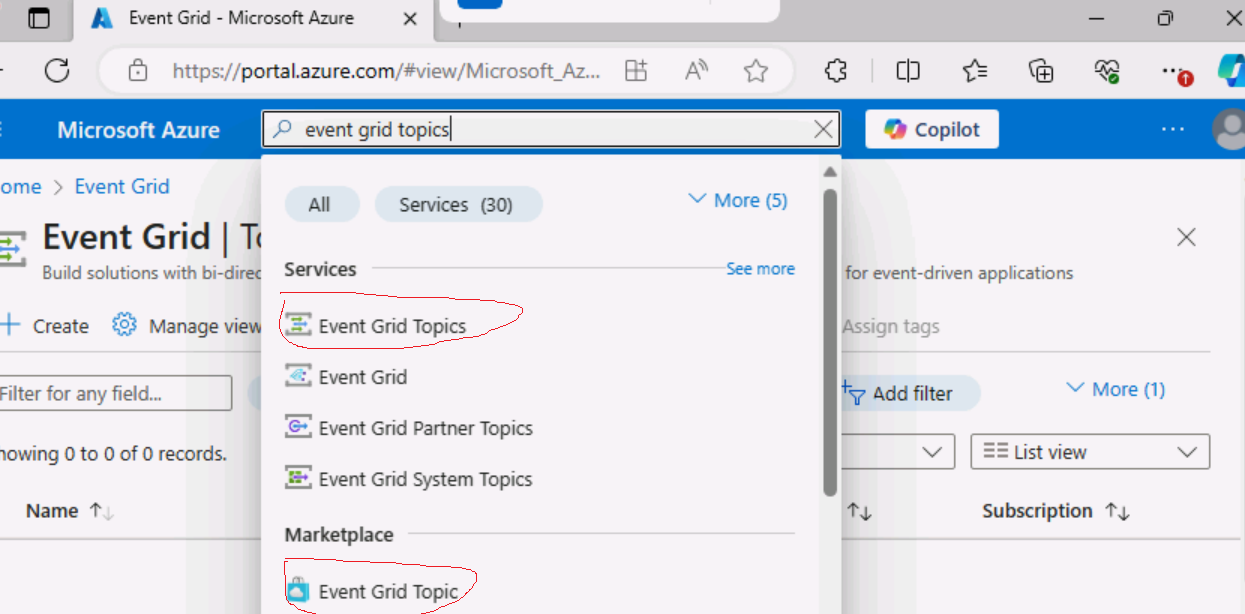
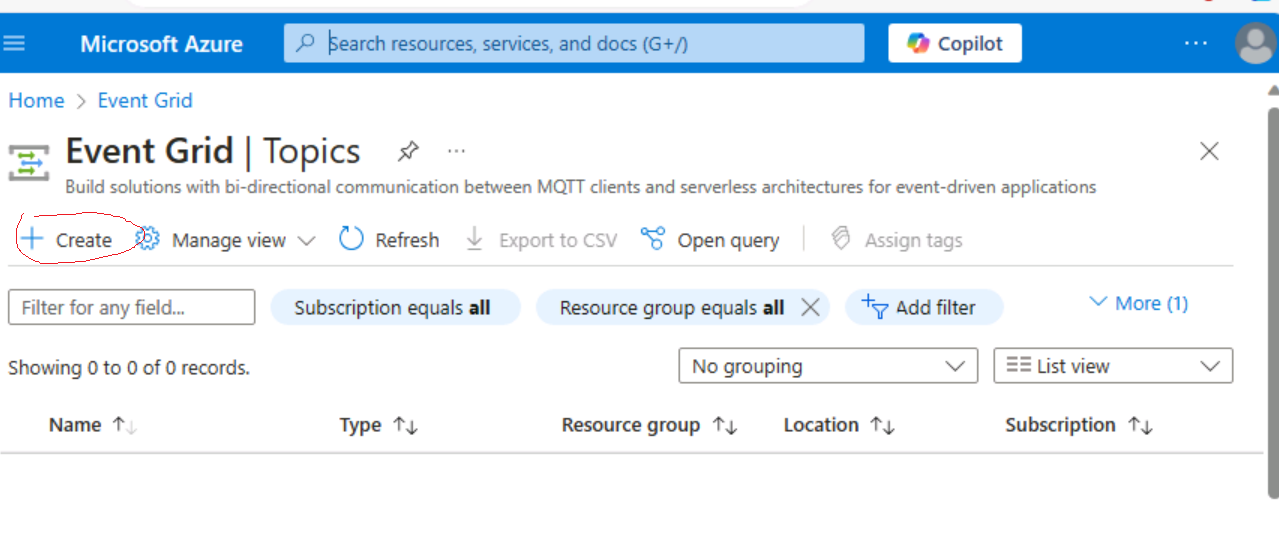
1. On the Azure portal's navigation pane, select **Create a resource**.

🡪On the **Create a resource** blade, in the **Search services and marketplace** text box, enter Event Grid Topic, and then select Enter.



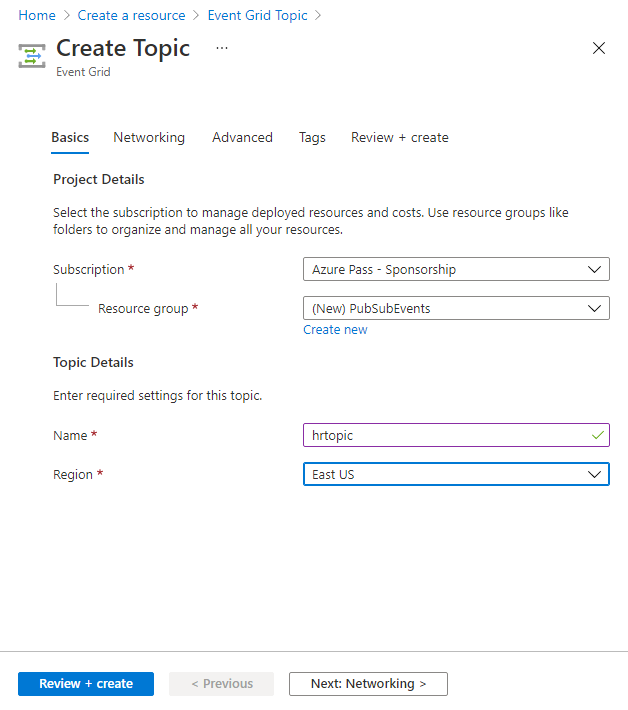
🡪On the **Marketplace** search results blade, select the **Event Grid Topic** result, and then select **Create**.

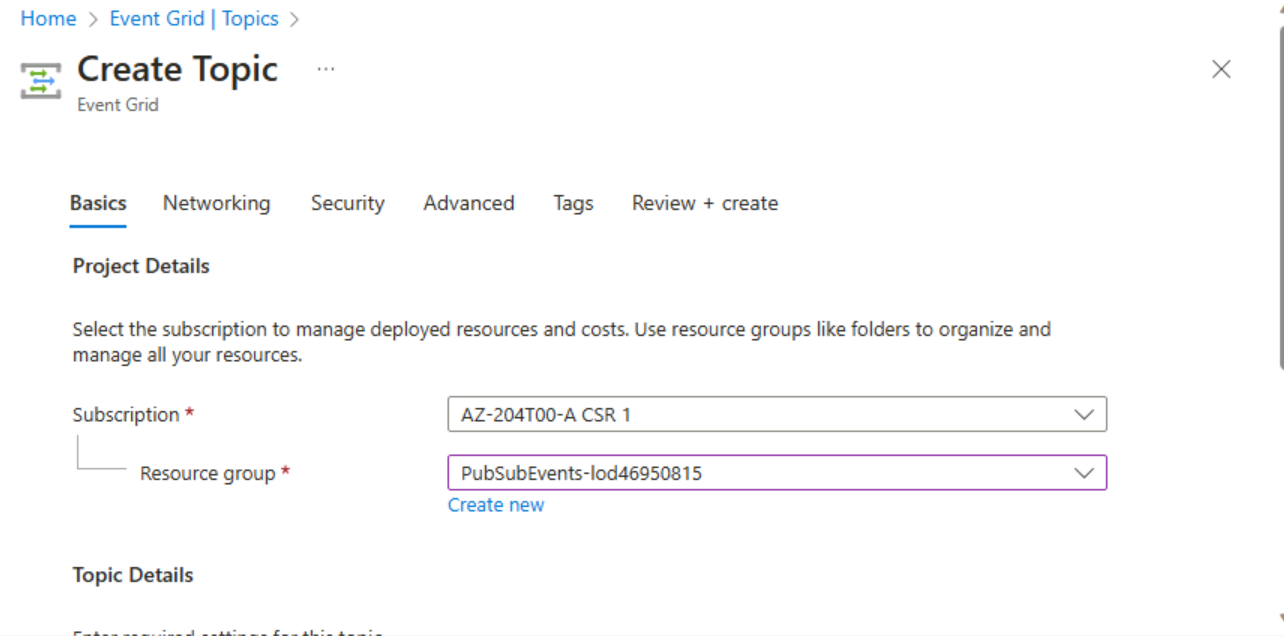


On the **Create Topic** blade, on the **Basics** tab, perform the following actions, and then select the **Advanced** tab:

| **Setting** | **Action** |
| --- | --- |
| **Subscription** drop-down list | Retain the default value |
| **Resource group** drop-down list | select the **PubSubEvents-lod46950815** resource group |
| **Name** text box | enter hrtopic46950815 |
| **Region** drop-down list | westus2 |

The following screenshot displays the configured settings on the **Basics** tab.

1. 



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

On the **Advanced** tab, from the **Event Schema** drop-down list, ensure that the **Event Grid Schema** entry is selected, and then select **Review + create**.

On the **Review + create** tab, review the options that you selected during the previous steps.

Select **Create** to create the event grid topic by using your specified configuration.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

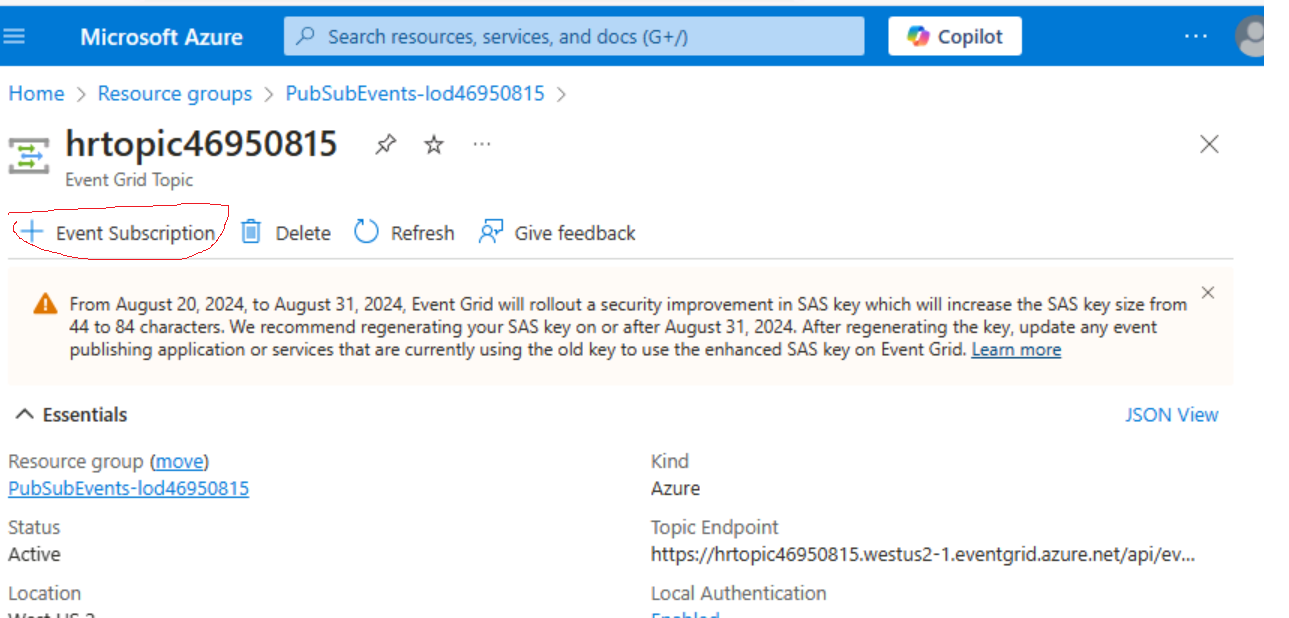
Description automatically generated

A screenshot of a computer

Description automatically generated

Exercise 2: Create an Event Grid subscription

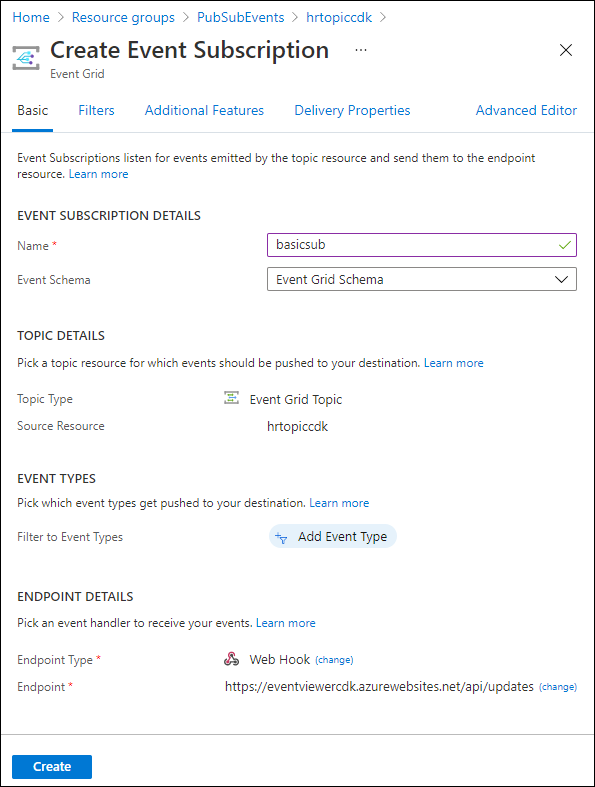
On the **Event Grid Topic** blade, select **+ Event Subscription**.



On the **Create Event Subscription** blade, perform the following actions, and then select **Create**:

| **Setting** | **Action** |
| --- | --- |
| **Name** text box | enter basicsub |
| **Event Schema** drop-down list | Select **Event Grid Schema** |
| **Endpoint Type** drop-down list | Select **Web Hook** |
| **Endpoint** | Select **Configure an endpoint**. In the **Subscriber Endpoint** text box, enter the **Web App URL** value that you recorded previously, ensure that it uses an **https://** prefix, add the suffix **/api/updates**, and then select **Confirm Selection**. For example, if your **Web App URL** value is http://eventviewerstudent.azurewebsites.net/, then your **Subscriber Endpoint** would be https://eventviewerstudent.azurewebsites.net/api/updates |

The following screenshot displays the configured settings on the **Create Event Subscription** blade.



**Note**: Wait for Azure to finish creating the subscription before you continue with the lab. You'll receive a notification when the subscription is created.

A screenshot of a web page

Description automatically generated

A screenshot of a web page

Description automatically generated

Task 4: Record subscription credentials

1. On the Azure portal's navigation pane, select **Resource groups**.
2. On the **Resource groups** blade, select the PubSubEvents-lod46950815 resource group that you created previously in this lab.
3. On the PubSubEvents-lod46950815 blade, select the **hrtopic46950815** Event Grid topic that you created previously in this lab.
4. On the **Event Grid Topic** blade, record the value of the **Topic Endpoint** field. You'll use this value later in the lab.
5. In the **Settings** category, select the **Access keys** link.
6. In the **Access keys** section, record the value of the **Key 1** text box. You'll use this value later in the lab.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

using Azure;

using Azure.Messaging.EventGrid;

using System;

using System.Threading.Tasks;

public class Program

{

    private const string topicEndpoint = "https://hrtopic46950815.westus2-1.eventgrid.azure.net/api/events";

    /\* Update the topicEndpoint string constant by setting its value to the Topic

       Endpoint of the Event Grid topic that you recorded previously in this lab. \*/

    private const string topicKey = "9XEZ6Vcj94qNayljVpNXSm2bFWcyK7UTxFcEAvtugjPohbfOuVxmJQQJ99ALAC8vTInXJ3w3AAABAZEGaDac";

    /\* Update the topicKey string constant by setting its value to the Key of the Event Grid topic that you recorded previously in this lab. \*/

    public static async Task Main(string[] args)

{

    /\* To create a new variable named "endpoint" of type "Uri",

       using the "topicEndpoint" string constant as a constructor parameter \*/

    Uri endpoint = new Uri(topicEndpoint);

    /\* To create a new variable named "credential" of type "AzureKeyCredential",

       use the "topicKey" string constant as a constructor parameter. \*/

    AzureKeyCredential credential = new AzureKeyCredential(topicKey);

    /\* To create a new variable named "client" of type "EventGridPublisherClient",

       using the "endpoint" and "credential" variables as constructor parameters \*/

    EventGridPublisherClient client = new EventGridPublisherClient(endpoint, credential);

    /\* To create a new variable named "firstEvent" of type "EventGridEvent",

       and populate that variable with sample data \*/

    EventGridEvent 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"

        }

    );

    /\* To create a new variable named "secondEvent" of type "EventGridEvent",

       and populate that variable with sample data \*/

    EventGridEvent 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"

        }

    );

    /\* To asynchronously invoke the "EventGridPublisherClient.SendEventAsync"

       method using the "firstEvent" variable as a parameter \*/

    await client.SendEventAsync(firstEvent);

    Console.WriteLine("First event published");

    /\* To asynchronously invoke the "EventGridPublisherClient.SendEventAsync"

       method using the "secondEvent" variable as a parameter \*/

    await client.SendEventAsync(secondEvent);

    Console.WriteLine("Second event published");

}

}