AZ-204

Developing solutions for Microsoft Azure

Lab 09

Publishing and subscribing to Event Grid events

Table of Contents

1 Pre-requisites

1.1 Sign in to the lab virtual machine

Sign in to your Windows 10 virtual machine (VM) by using the following credentials:

- Username: **Admin**
- Password: **Pa55w.rd**

Note: Instructions to connect to the virtual lab environment will be provided by your instructor.

1.2 Review the installed applications

Find the taskbar on your Windows 10 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

- Microsoft Edge
- File Explorer
- Azure CLI
- Windows PowerShell

2 Exercise 1: Create Azure resources

2.1 Task 1: Open the Azure portal

1. On the taskbar, select the **Microsoft Edge** icon.
2. In the open browser window, go to the Azure portal (https://portal.azure.com).
3. Enter the email address for your Microsoft account, and then select **Next**.
4. Enter the password for your Microsoft account, and then select **Sign in**.

   Note: If this is your first time signing in to the Azure portal, you'll be offered a tour of the portal. Select **Get Started** to skip the tour and begin using the portal.

2.2 Task 2: Open Azure Cloud Shell

1. In the Azure portal, select the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is represented by a greater than sign (>) and underscore character (_).

2. If this is your first time opening Cloud Shell using your subscription, you can use the **Welcome to Azure Cloud Shell Wizard** to configure Cloud Shell. Perform the following actions in the wizard:
   - When a dialog box prompts you to create a new storage account to begin using the shell, accept the default settings, and then select **Create storage**.

   Note: Wait for Cloud Shell to finish its initial setup procedures before continuing with the lab. If you don't notice the **Cloud Shell** configuration options, this is most likely because you're using an existing subscription with this course's labs. The labs are written with the presumption that you're using a new subscription.

3. In Azure portal, at the **Cloud Shell** command prompt enter the following command, and then select Enter to get the version of the Azure Command-Line Interface (Azure CLI) tool:

   ```
   az --version
   ```

2.3 Task 3: View the Microsoft.EventGrid provider registration

1. At the **Cloud Shell** command prompt in the portal, perform the following actions:
   1. Enter the following command, and then select Enter to get a list of subgroups and commands at the root level of the Azure CLI:

      ```
      az --help
      ```

   2. Enter the following command, and then select Enter to get a list of the commands that are available for resource providers:

      ```
      az provider --help
      ```

   3. Enter the following command, and then select Enter to list all currently registered providers:

      ```
      az provider list
      ```

4. Enter the following command, and then select Enter to list just the namespaces of the currently registered providers:

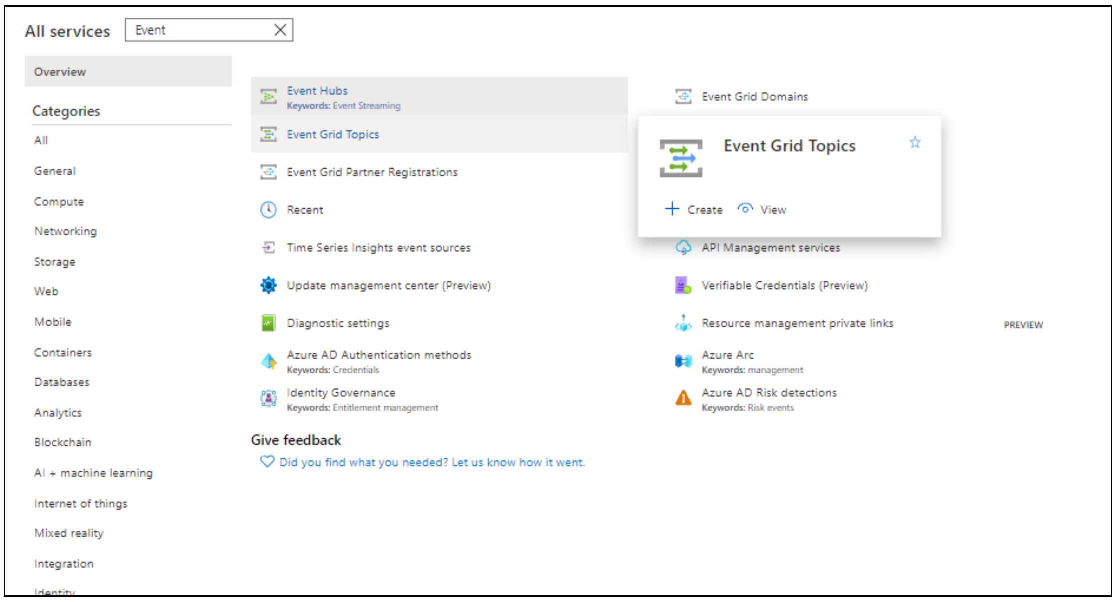
```
az provider list --query "[].namespace"
```

4. Review the list of currently registered providers. Notice that the **Microsoft.EventGrid** provider is currently included in the list of providers.

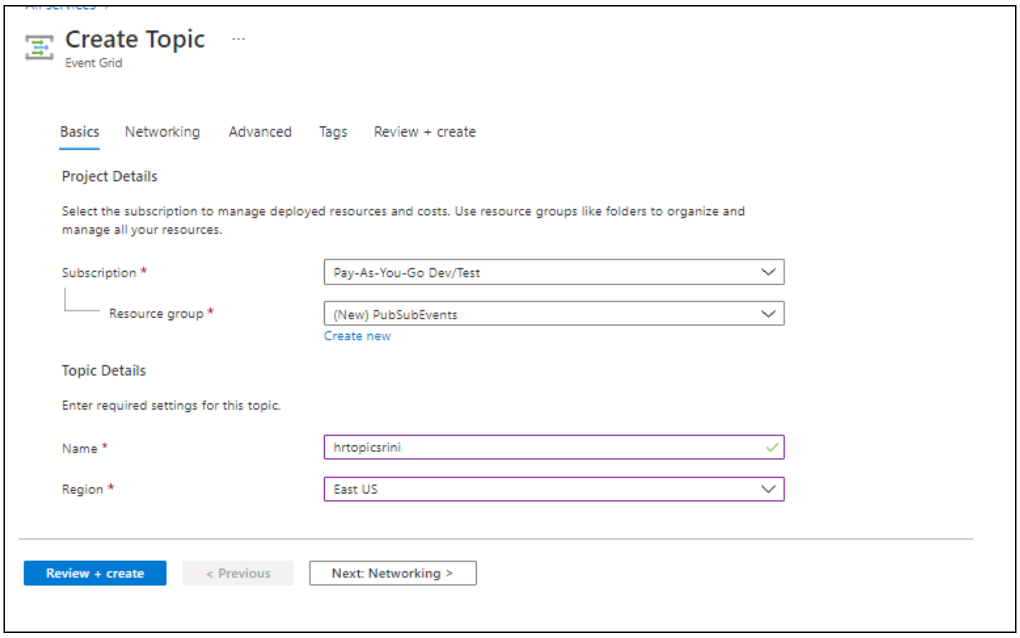
2. Close the Cloud Shell pane.

2.4 Task 4: Create a custom Event Grid topic

1. In the Azure portal's navigation pane, select **Create a resource**.
2. On the **Create a resource** blade, find the **Search services and marketplace** text box.
3. In the search box, enter **Event Grid Topic**, and then select Enter.
4. From the **Marketplace** search results blade, select the **Event Grid Topic** result.



5. From the **Event Grid Topic** blade, select **Create**.
6. From the **Create Topic** blade, perform the following actions:
   1. In the **Name** text box, enter **hrtopic*[yourname]***.
   2. In the **Resource group** section, select **Create new**, enter **PubSubEvents**, and then select **OK**.
   3. From the **Region** drop-down list, select the **(US) East US** region.


Create Topic ···
Event Grid

Basics | Networking | Advanced | Tags | Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Pay-As-You-Go Dev/Test
Resource group *	(New) PubSubEvents

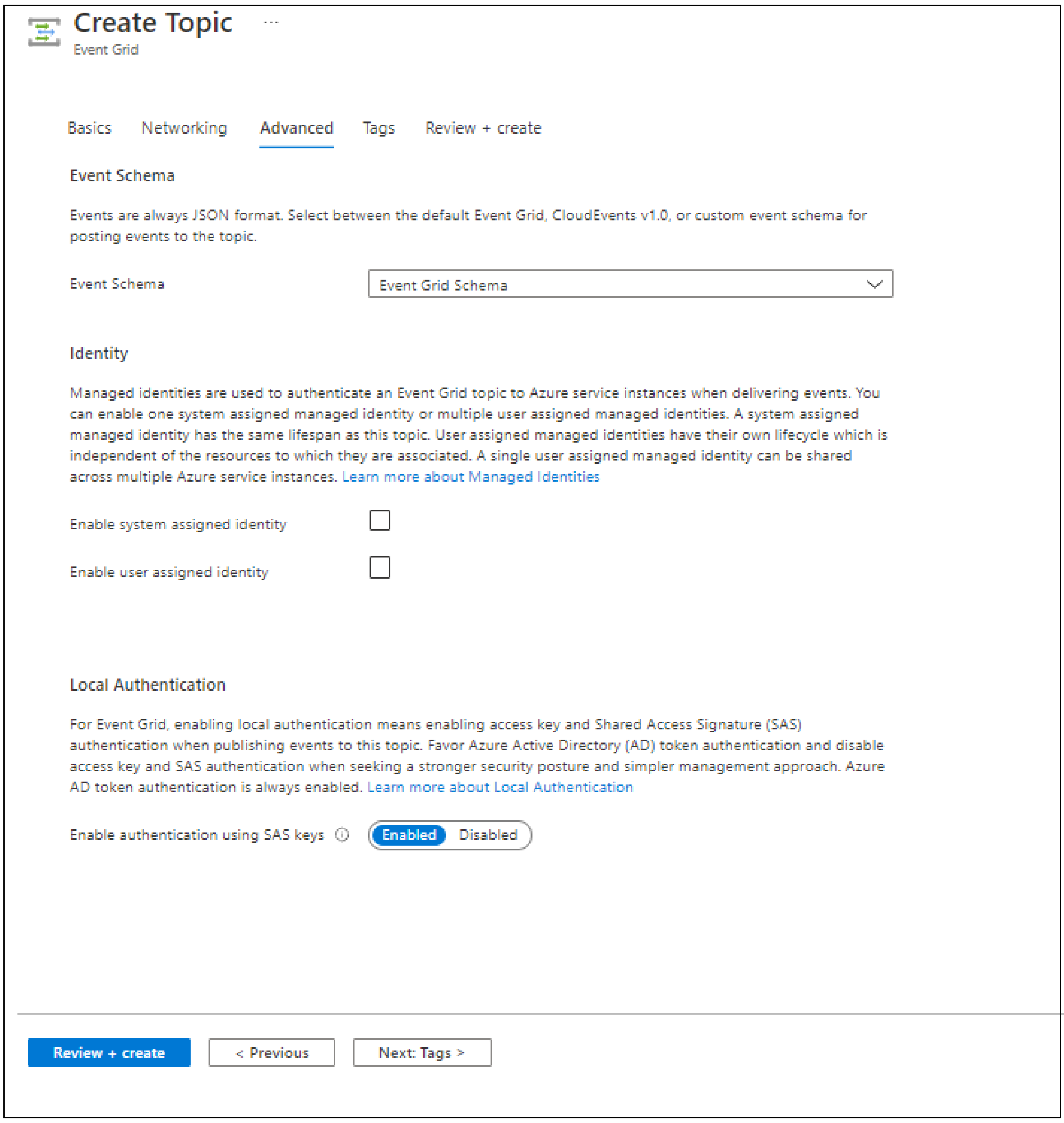
Create new

Topic Details

Enter required settings for this topic.

Name *	hrtopicsrini
Region *	East US

4. Select the **Advanced** tab.
5. From the **Event Schema** drop-down list, select **Event Grid Schema**.


Create Topic ...
Event Grid

Basics | Networking | Advanced | Tags | Review + create

Event Schema

Events are always JSON format. Select between the default Event Grid, CloudEvents v1.0, or custom event schema for posting events to the topic.

Event Schema: Event Grid Schema

Identity

Managed identities are used to authenticate an Event Grid topic to Azure service instances when delivering events. You can enable one system assigned managed identity or multiple user assigned managed identities. A system assigned managed identity has the same lifespan as this topic. User assigned managed identities have their own lifecycle which is independent of the resources to which they are associated. A single user assigned managed identity can be shared across multiple Azure service instances. Learn more about Managed Identities

Enable system assigned identity ☐

Enable user assigned identity ☐

Local Authentication

For Event Grid, enabling local authentication means enabling access key and Shared Access Signature (SAS) authentication when publishing events to this topic. Favor Azure Active Directory (AD) token authentication and disable access key and SAS authentication when seeking a stronger security posture and simpler management approach. Azure AD token authentication is always enabled. Learn more about Local Authentication

Enable authentication using SAS keys: [✓ Enabled] Disabled

6. Select **Review + Create**.
7. On the **Review + Create** tab, review the options that you selected during the previous steps.
8. Select **Create** to create the event grid topic by using your specified configuration.

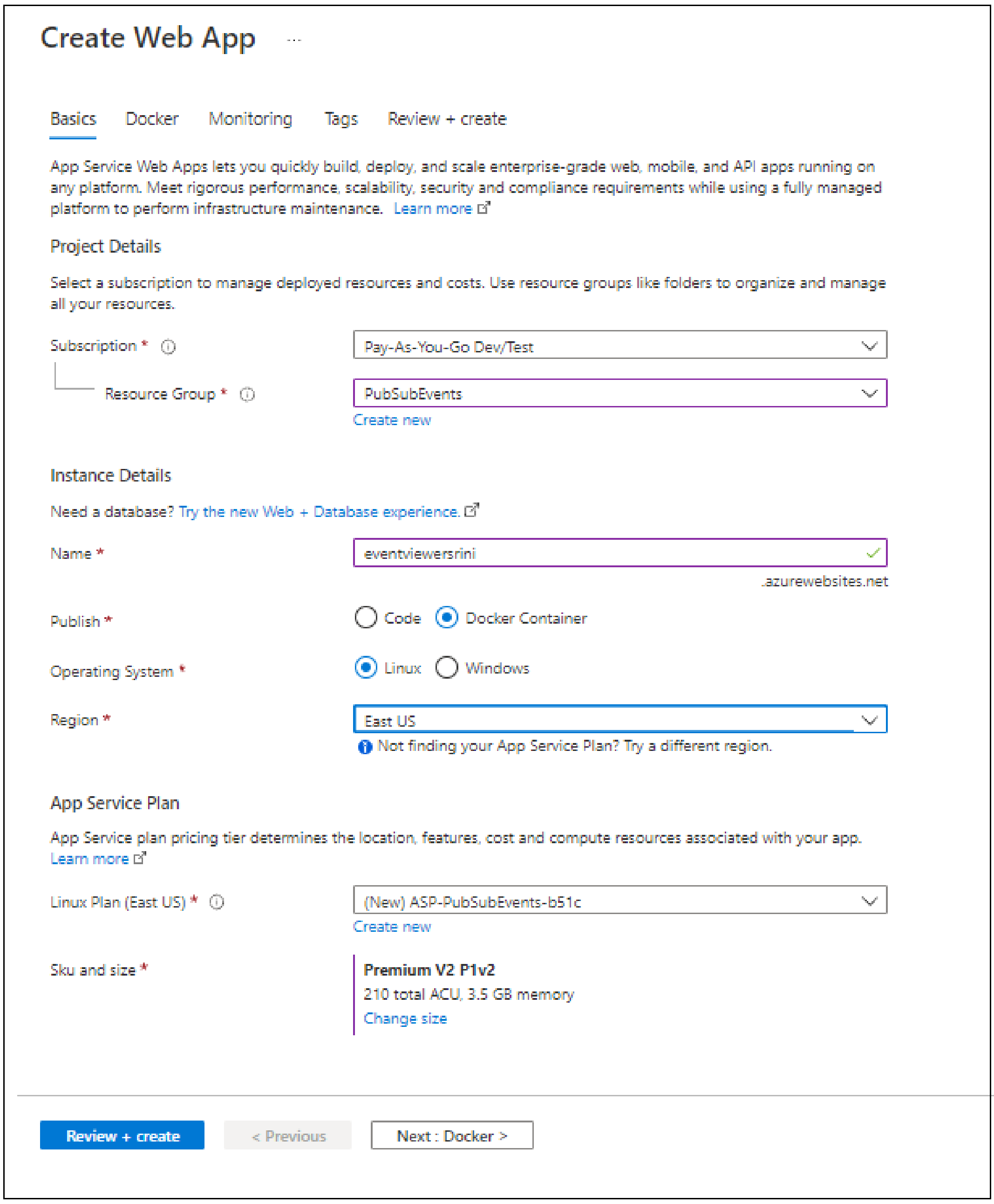
   Note: Wait for Azure to finish creating the topic before you continue with the lab. You'll receive a notification when the topic is created.

2.5 Task 5: Deploy the Azure Event Grid viewer to a web app

1. In the Azure portal's navigation pane, select **Create a resource**.
2. On the **Create a resource** blade, find the **Search services and marketplace** text box.
3. In the search box, enter **App Service**, and then select Enter.
4. From the **Marketplace** search results blade, select the **App Service** result.
5. From the **Web App** blade, select **Create**.
6. From the **Create Web App** blade, find the tabs on the blade, such as **Basics**.

Note: Each tab represents a step in the workflow to create a new web app. You can select **Review + Create** at any time to skip the remaining tabs.

7. On the **Basics** tab, perform the following actions:
    1. Leave the **Subscription** text box set to its default value.
    2. In the **Resource group** section, select **PubSubEvents**.
    3. In the **Name** text box, enter **eventviewer*[yourname]***.
    4. In the **Publish** section, select **Docker Container**.
    5. In the **Operating System** section, select **Linux**.
    6. From the **Region** drop-down list, select the **East US** region.
    7. In the **Linux Plan (East US)** section, select **Create new**.
    8. In the **Name** text box, enter the value **EventPlan**, and then select **OK**.
    9. Leave the **SKU and size** section set to its default value.


Create Web App

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Pay-As-You-Go Dev/Test
Resource Group * : PubSubEvents
Create new

Instance Details

Need a database? Try the new Web + Database experience.

Name * : eventviewersrini .azurewebsites.net
Publish * : Code [✓ Docker Container]
Operating System * : [✓ Linux] Windows
Region * : East US
Not finding your App Service Plan? Try a different region.

App Service Plan

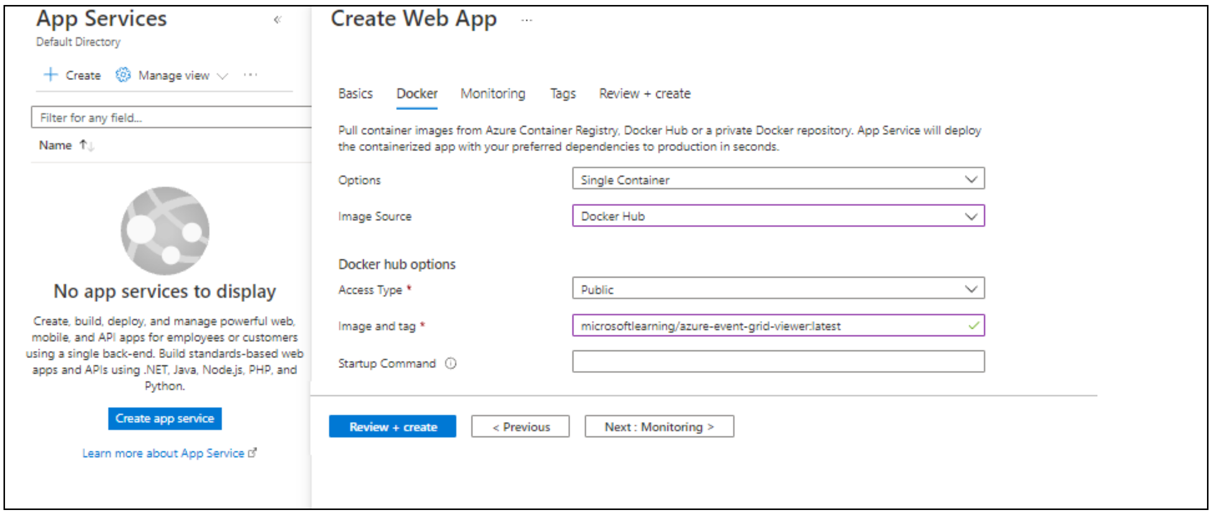
App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. Learn more

Linux Plan (East US) * : (New) ASP-PubSubEvents-b51c
Create new

Sku and size * : **Premium V2 P1v2**
210 total ACU, 3.5 GB memory
Change size

10. Select **Next: Docker**.

8. On the **Docker** tab, perform the following actions:
    1. From the **Options** drop-down list, select **Single Container**.
    2. From the **Image Source** drop-down list, select **Docker Hub**.
    3. From the **Access Type** drop-down list, select **Public**.
    4. In the **Image and tag** text box, enter **microsoftlearning/azure-event-grid-viewer:latest**.



Create, build, deploy, and manage powerful web, mobile, and API apps for employees or customers using a single back-end. Build standards-based web apps and APIs using .NET, Java, Node.js, PHP, and Python.

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options: Single Container
Image Source: Docker Hub

Docker hub options
Access Type *: Public
Image and tag *: microsoftlearning/azure-event-grid-viewer:latest
Startup Command:

    5. Select **Review + Create**.

9. On the **Review + Create** tab, review the options that you selected during the previous steps.
10. Select **Create** to create the web app using your specified configuration.

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

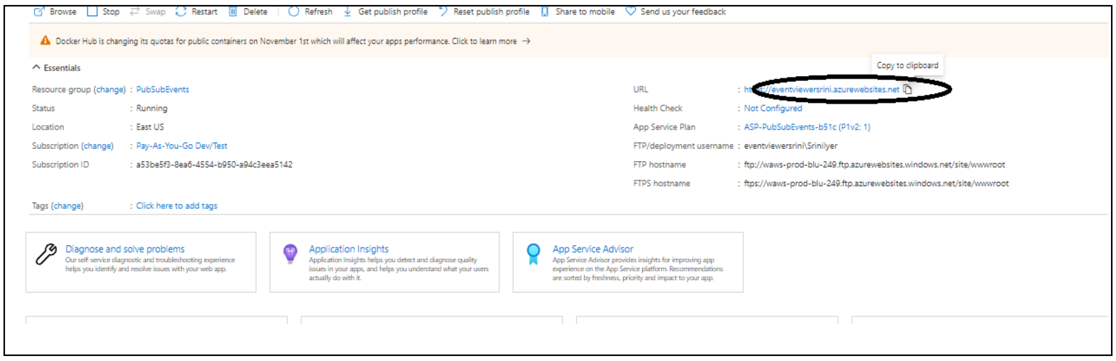
2.6 Review

In this exercise, you created the Event Grid topic and a web app that you will use throughout the remainder of the lab.

3 Exercise 2: Create an Event Grid subscription

3.1 Task 1: Access the Event Grid Viewer web application

1. In the Azure portal’s navigation pane, select **Resource groups**.
2. On the **Resource groups** blade, select the **PubSubEvents** resource group that you created earlier in this lab.
3. On the **PubSubEvents** blade, select the **eventviewer*[yourname]*** web app that you created earlier in this lab.
4. On the **App Service** blade, in the **Settings** category, select the **Properties** link.
5. In the **Properties** section, record the value of the **URL** text box. You’ll use this value later in the lab.



Docker Hub is changing its quotas for public containers on November 1st which will affect your apps performance. Click to learn more →

Essentials

Copy to clipboard

Resource group (change)	: PubSubEvents
Status	: Running
Location	: East US
Subscription (change)	: Pay-As-You-Go Dev/Test
Subscription ID	: a53be5f3-8ea6-4554-b950-a94c3eea5142
Tags (change)	: Click here to add tags

URL	: https://eventviewersrini.azurewebsites.net
Health Check	: Not Configured
App Service Plan	: ASP-PubSubEvents-b51c (P1v2: 1)
FTP/deployment username	: eventviewersrini\SriniIyer
FTP hostname	: ftp://waws-prod-blu-249.ftp.azurewebsites.windows.net/site/wwwroot
FTPS hostname	: ftps://waws-prod-blu-249.ftp.azurewebsites.windows.net/site/wwwroot

Diagnose and solve problems
Our self-service diagnostic and troubleshooting experience helps you identify and resolve issues with your web app.

Application Insights
Application Insights helps you detect and diagnose quality issues in your apps, and helps you understand what your users actually do with it.

App Service Advisor
App Service Advisor provides insights for improving app experience on the App Service platform. Recommendations are sorted by freshness, priority and impact to your app.

6. Select **Overview**.
7. In the **Overview** section, select **Browse**.
8. Observe the currently running **Azure Event Grid viewer** web application. Leave this web application running for the remainder of the lab.

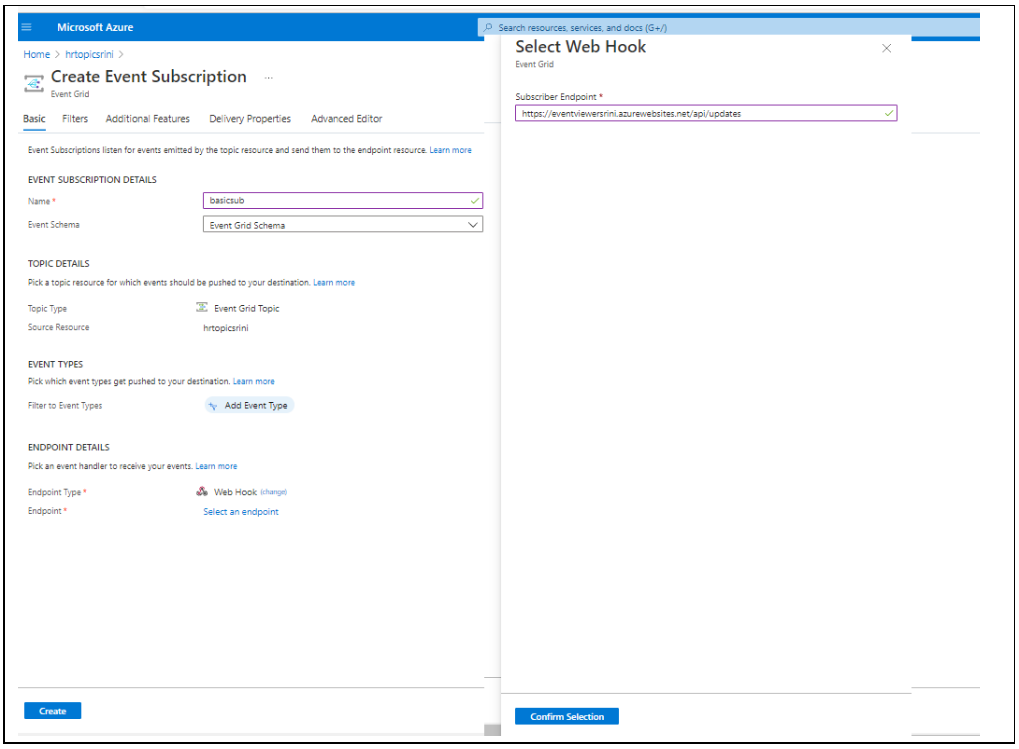
   Note: This web application will update in real-time as events are sent to its endpoint. We will use this to monitor events throughout the lab.

9. Return to your currently open browser window that’s displaying the Azure portal.

3.2 Task 2: Create new subscription

1. In the Azure portal’s navigation pane, select **Resource groups**.
2. On the **Resource groups** blade, select the **PubSubEvents** resource group that you created earlier in this lab.
3. On the **PubSubEvents** blade, select the **hrtopic*[yourname]*** Event Grid topic that you created earlier in this lab.
4. On the **Event Grid Topic** blade, select **+ Event Subscription**.
5. On the **Create Event Subscription** blade, perform the following actions:
   1. In the **Name** text box, enter **basicsub**.
   2. In the **Event Schema** list, select **Event Grid Schema**.
   3. In the **Endpoint Type** list, select **Web Hook**.
   4. Select **Select an endpoint**.
   5. In the **Select Web Hook** dialog box, in the **Subscriber Endpoint** text box, enter the **Web App URL** value that you recorded earlier, ensure it uses an **https://** prefix, add the suffix **/api/updates**, and then select **Confirm Selection**.

Note: 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.



Create Event Subscription

Event Grid

Event Subscriptions listen for events emitted by the topic resource and send them to the endpoint resource. Learn more

EVENT SUBSCRIPTION DETAILS

Name * basicsub

Event Schema Event Grid Schema

TOPIC DETAILS

Pick a topic resource for which events should be pushed to your destination. Learn more

Topic Type Event Grid Topic

Source Resource hrtopicsrini

EVENT TYPES

Pick which event types get pushed to your destination. Learn more

Filter to Event Types

ENDPOINT DETAILS

Pick an event handler to receive your events. Learn more

Endpoint Type * Web Hook (change)

Endpoint * Select an endpoint

Select Web Hook

Event Grid

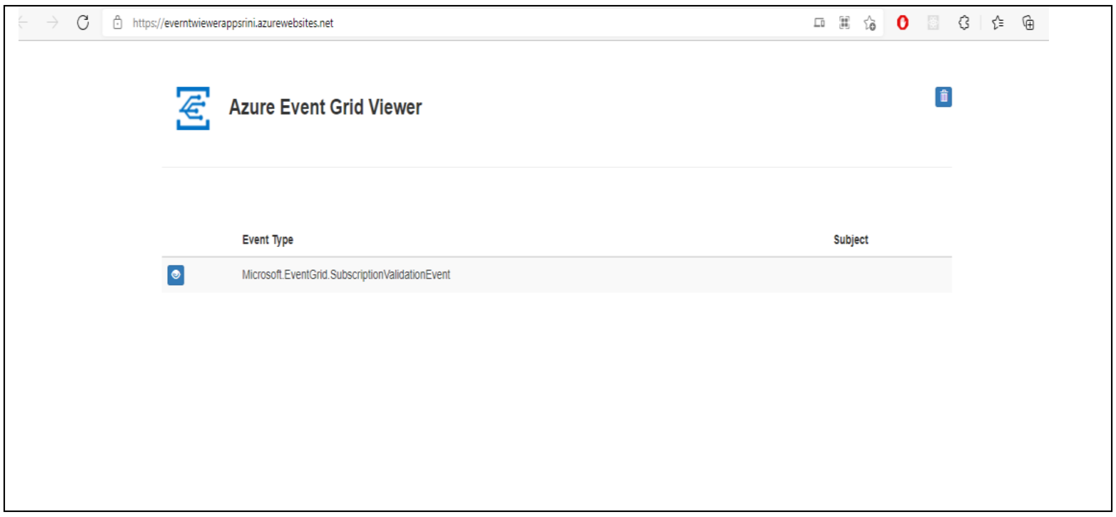
Subscriber Endpoint * https://eventviewersrini.azurewebsites.net/api/updates

6. Select **Create**.

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.

3.3 Task 3: Observe the subscription validation event

1. Return to the browser window displaying the **Azure Event Grid viewer** web application.
2. Review the **Microsoft.EventGrid.SubscriptionValidationEvent** event that was created as part of the subscription creation process.
3. Select the event and review its JSON content.


https://everntwiewerappsrini.azurewebsites.net

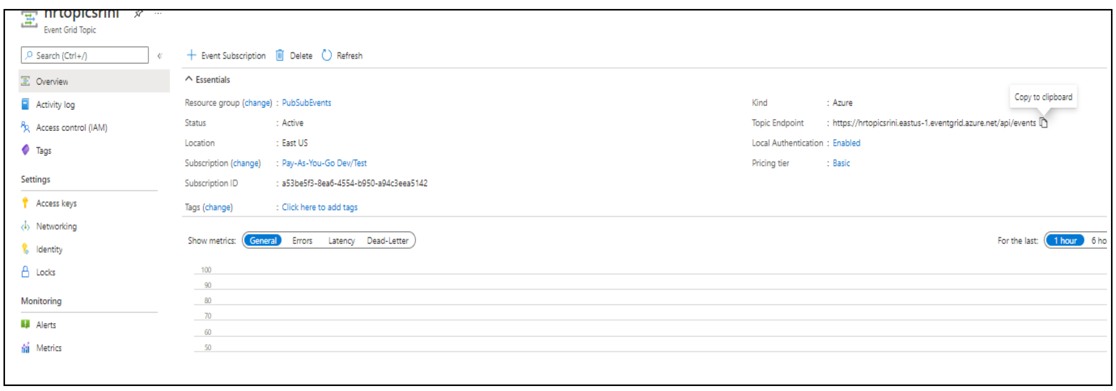
Azure Event Grid Viewer

Event Type	Subject
Microsoft.EventGrid.SubscriptionValidationEvent	

4. Return to your currently open browser window with the Azure portal.

3.4 Task 4: Record subscription credentials

1. In the Azure portal's navigation pane, select **Resource groups**.
2. On the **Resource groups** blade, select the **PubSubEvents** resource group that you created earlier in this lab.
3. On the **PubSubEvents** blade, select the **hrtopic*[yourname]*** Event Grid topic that you created earlier 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.

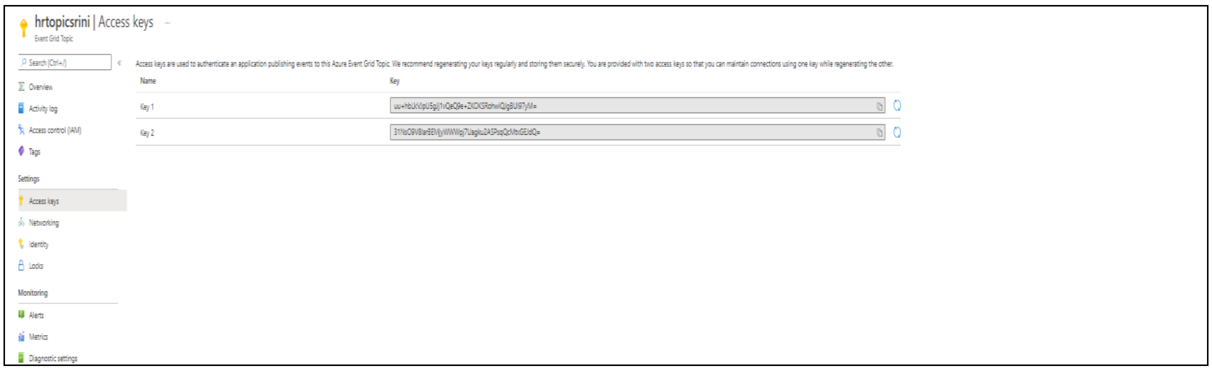

hrtopicsrini
Event Grid Topic

Essentials

Resource group (change) : PubSubEvents
Status : Active
Location : East US
Subscription (change) : Pay-As-You-Go Dev/Test
Subscription ID : a53be5f3-8ea6-4554-b950-a94c3eea5142
Tags (change) : Click here to add tags

Kind : Azure
Topic Endpoint : https://hrtopicsrini.eastus-1.eventgrid.azure.net/api/events
Local Authentication : Enabled
Pricing tier : Basic

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.



Access keys are used to authenticate an application publishing events to this Azure Event Grid Topic. We recommend regenerating your keys regularly and storing them securely. You are provided with two access keys so that you can maintain connections using one key while regenerating the other.

Name	Key
Key 1	uu+hbUKVIpU5gJj1vQeQ9e+ZKOKSRohwiQJgBUI97yM=
Key 2	31NsO9V8larBEMjyWWWpj7Uagku2ASPsqQcMtxGEJdQ=

3.5 Review

In this exercise, you created a new subscription, validated its registration, and then recorded the credentials required to publish a new event to the topic.

4 Exercise 3: Publish Event Grid events from .NET

4.1 Task 1: Create a .NET project

1. On the **Start** screen, select the **Visual Studio Code** tile.
2. From the **File** menu, select **Open Folder**.
3. In the **File Explorer** window that opens, browse to **Allfiles (F):\Allfiles\Labs\09\Starter\EventPublisher**, and then select **Select Folder**.
4. In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.
5. At the open command prompt, enter the following command, and then select Enter to create a new .NET project named **EventPublisher** in the current folder:

```
dotnet new console --name EventPublisher --output .
```

Note: The **dotnet new** command will create a new **console** project in a folder with the same name as the project.

6. At the command prompt, enter the following command, and then select Enter to import version 4.1.0 of **Azure.Messaging.EventGrid** from NuGet:

```
dotnet add package Azure.Messaging.EventGrid --version 4.1.0
```

Note: The **dotnet add package** command will add the **Microsoft.Azure.EventGrid** package from NuGet. For more information, go to Azure.Messaging.EventGrid.

At the command prompt, enter the following command, and then select Enter to build the .NET web application:

```
dotnet build
```

7. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

4.2 Task 2: Modify the Program class to connect to Event Grid

1. In the Explorer pane of the **Visual Studio Code** window, open the **Program.cs** file.
2. On the code editor tab for the **Program.cs** file, delete all the code in the existing file.
3. Add the following line of code to import the **Azure**, and **Azure.Messaging.EventGrid** namespaces from the **Azure.Messaging.EventGrid** package imported from NuGet:

Add the following lines of code to add **using** directives for the

```
using Azure;
using Azure.Messaging.EventGrid;

using System;
using System.Threading.Tasks;
```

4. In the **Program** class, enter the following line of code to create a new string constants named **topicEndpoint** & **topicKey**

5. Update the **topicEndpoint and topicKey** string constants by values that we have recorded
6. Update the Program.cs as below with an async Main method.

```
using Azure;
using Azure.Messaging.EventGrid;

using System;
using System.Threading.Tasks;



namespace EventPublisher
{
    class Program
    {
        private const string topicEndpoint = "https://hrtopicsrini.eastus-1.eventgrid.azure.net/api/events";
        private const string topicKey = "uu+hbLkVJpU5gJj1vQeQ9e+ZKOKSRohwiQJgBUl97yM=";

        public static async Task Main(string[] args)
        {

        }

    }
}
```

4.3 Task 3: Publish new events

1. In the **Main** method, perform the following actions to publish a list of events to your topic endpoint:
    1. Add the following line of code to create a new variable named **endpoint** of type **Uri**, using the **topicEndpoint** string constant as a constructor parameter:

```
Uri endpoint = new Uri(topicEndpoint);
```

    2. Add the following line of code to create a new variable named **credential** of type AzureKeyCredential, using the **topicKey** string constant as a constructor parameter:

```
 AzureKeyCredential credential = new AzureKeyCredential(topicKey);
```

    3. Add the following line of code 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);
```

4. Add the following block of code 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"
                            }
                    );
```

5. Add the following block of code 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"
                       }
                   );
```

6. Add the following line of code to asynchronously invoke the EventGridPublisherClient.SendEventAsync method using the **firstEvent** variable as a parameter:

```
 await client.SendEventAsync(firstEvent);
```

7. Add the following line of code to render the **"First event published"** message to the console:

```
 Console.Out.WriteLineAsync("First event published");
```

8. Add the following line of code to asynchronously invoke the EventGridPublisherClient.SendEventAsync method using the **secondEvent** variable as a parameter:

```
await client.SendEventAsync(secondEvent);
```

9. Add the following line of code to render the **“Second event published”** message to the console:

```
Console.Out.WriteLineAsync("Second event published");
```

10. Review the **Program.cs** file:

```
using Azure;
using Azure.Messaging.EventGrid;

using System;
using System.Threading.Tasks;



namespace EventPublisher
{
    class Program
    {
        private const string topicEndpoint = "https://hrtopicsrini.eastus-
1.eventgrid.azure.net/api/events";
        private const string topicKey = "uu+hbLkVJpU5gJj1vQeQ9e+ZKOKSRohwiQJgBUl97yM=";

        public static async Task Main(string[] args)
        {
                Uri endpoint = new Uri(topicEndpoint);
                AzureKeyCredential credential = new AzureKeyCredential(topicKey);

                EventGridPublisherClient client = new EventGridPublisherClient(endpoint
, credential);
                   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"
                            }
                    );
                    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"
                        }
                    );
                await client.SendEventAsync(firstEvent);
                Console.Out.WriteLineAsync("First event published");
                await client.SendEventAsync(secondEvent);
                Console.Out.WriteLineAsync("Second event published");


        }
    }
}
2.
```

3. Save the Program.cs file.

In the **Visual Studio Code** window, right-click or activate the shortcut menu for the Explorer pane, and then select **Open in Terminal**.

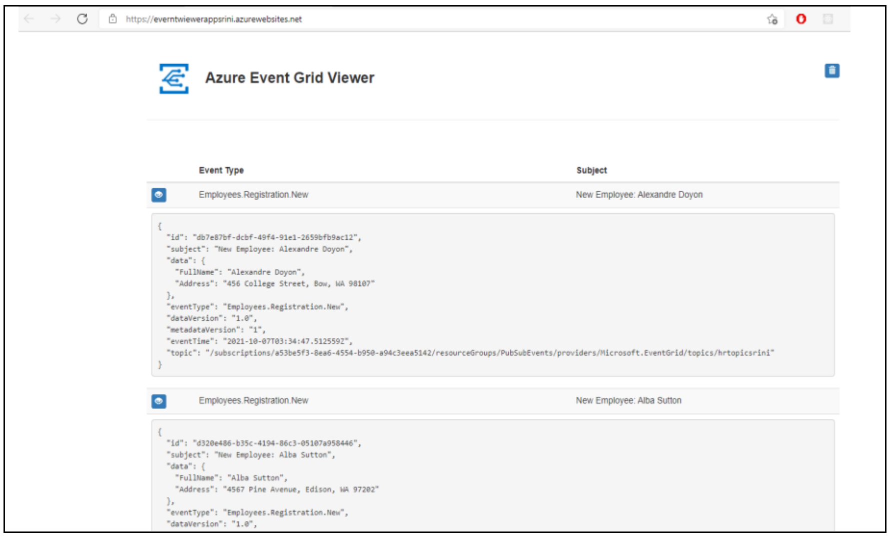
At the open command prompt, enter the following command, and then select Enter to run the .NET web application:

```
dotnet run
```

5. **Note**: If there are any build errors, review the **Program.cs** file in the **Allfiles (F):\Allfiles\Labs\09\Solution\EventPublisher** folder.
6. Observe the success message output from the currently running console application.
7. Select **Kill Terminal** or the **Recycle Bin** icon to close the currently open terminal and any associated processes.

4.4 Task 4: Observe published events

1. Return to the browser window with the **Azure Event Grid viewer** web application.
2. Review the **Employees.Registration.New** events that were created by your console application.
3. Select any of the events and review its JSON content.



Event Type	Subject
Employees.Registration.New	New Employee: Alexandre Doyon

```
{
  "id": "db7e87bf-dcbf-49f4-91e1-2659bfb9ac12",
  "subject": "New Employee: Alexandre Doyon",
  "data": {
    "FullName": "Alexandre Doyon",
    "Address": "456 College Street, Bow, WA 98107"
  },
  "eventType": "Employees.Registration.New",
  "dataVersion": "1.0",
  "metadataVersion": "1",
  "eventTime": "2021-10-07T03:34:47.512559Z",
  "topic": "/subscriptions/a53be5f3-8ea6-4554-b950-a94c3eea5142/resourceGroups/PubSubEvents/providers/Microsoft.EventGrid/topics/hrtopicsrini"
}
```

Event Type	Subject
Employees.Registration.New	New Employee: Alba Sutton

```
{
  "id": "d320e486-b35c-4194-86c3-05107a958446",
  "subject": "New Employee: Alba Sutton",
  "data": {
    "FullName": "Alba Sutton",
    "Address": "4567 Pine Avenue, Edison, WA 97202"
  },
  "eventType": "Employees.Registration.New",
  "dataVersion": "1.0",
```

4. Return to the Azure portal.

4.5 Review

In this exercise, you published new events to your Event Grid topic using a .NET console application.

5 Exercise 4: Clean up your subscription

5.1 Task 1: Open Azure Cloud Shell

1. In Azure portal, select the **Cloud Shell** icon to open a new shell instance.

   Note: The **Cloud Shell** icon is represented by a greater than sign (>) and underscore character (_).

2. If this is your first time opening Cloud Shell using your subscription, you can use the **Welcome to Azure Cloud Shell Wizard** to configure Cloud Shell for first-time usage. Perform the following actions in the wizard:
   1. A dialog box prompts you to configure the shell. Select **Bash**, review the selected subscription, and then select **Create storage**.

   Note: Wait for Cloud Shell to finish its initial setup procedures before moving forward with the lab. If you don't notice the Cloud Shell configuration options, this is most likely because you're using an existing subscription with this course's labs. The labs are written with the presumption that you're using a new subscription.

5.2 Task 2: Delete resource groups

1. Enter the following command, and then select Enter to delete the **PubSubEvents** resource group:

   ```
   az group delete --name PubSubEvents --no-wait --yes
   ```

2. Close the Cloud Shell pane in the portal.

5.3 Task 3: Close the active applications

1. Close the currently running Microsoft Edge application.
2. Close the currently running Visual Studio Code application.

5.4 Review

In this exercise, you cleaned up your subscription by removing the resource groups used in this lab.