Microsoft

AZ-204T00A

Learning Path 09: Develop event-based solutions

Agenda

- Explore Azure Event Grid
- Explore Azure Event Hubs

Module 1: Explore Azure Event Grid

Learning objectives

- Describe how Event Grid operates and how it connects to services and event handlers.
- Explain how Event Grid delivers events and how it handles errors.
- Implement authentication and authorization.
- Route custom events to web endpoint by using Azure CLI.

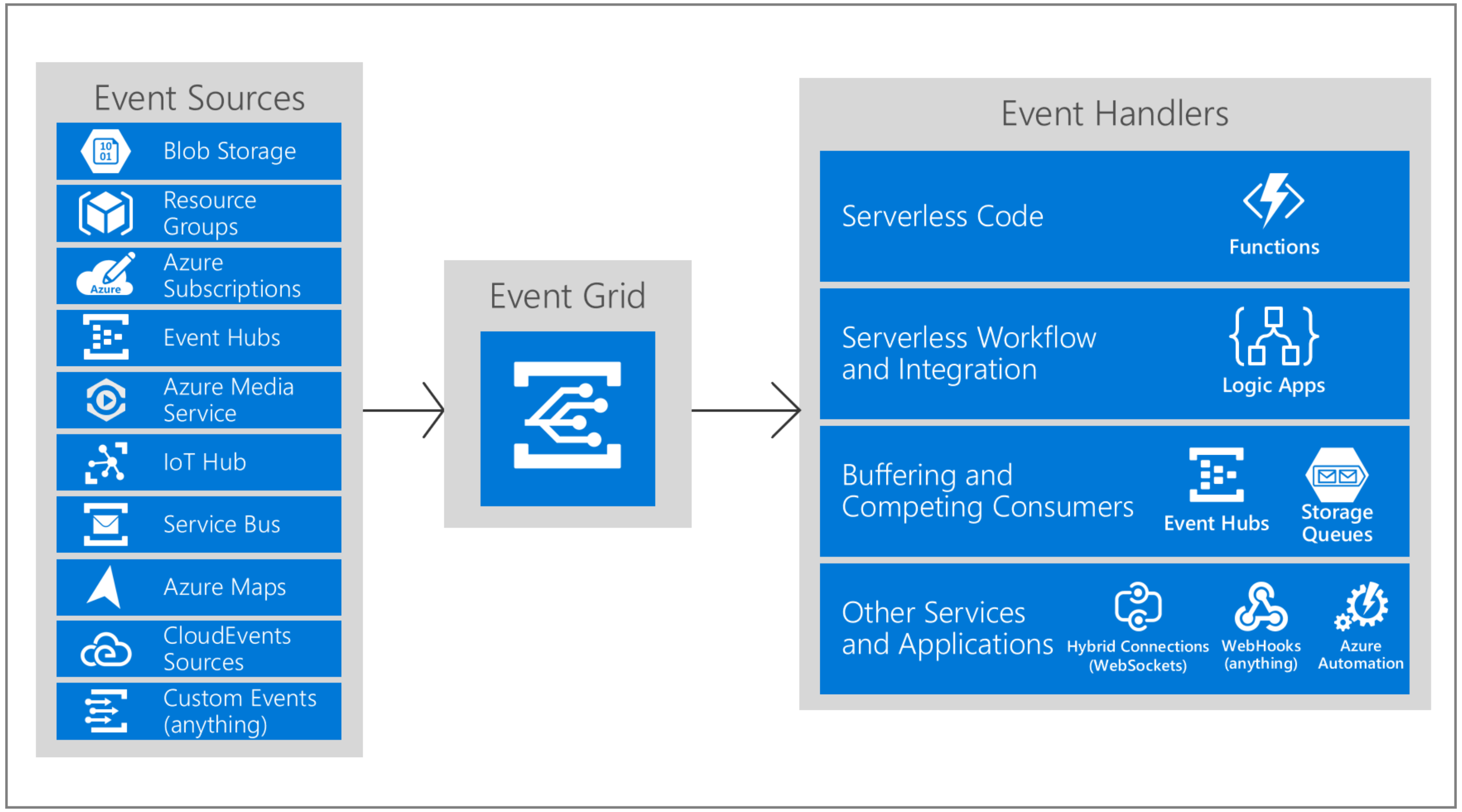
Introduction

- Azure Event Grid is deeply integrated with Azure services and can be integrated with third-party services.
- It simplifies event consumption and lowers costs by eliminating the need for constant polling.
- Event Grid efficiently and reliably routes events from Azure, and non-Azure resources, and distributes the events to registered subscriber endpoints.

Explore Azure Event Grid (1 of 2)

What is the event grid

- Azure Event Grid is an eventing backplane
- Built-in support for events coming from Azure services
- Create events with custom topics
- Use filters to route specific events to different endpoints



Explore Azure Event Grid (2 of 2)

Five key concepts

1. **Events** - An event is the smallest amount of information that fully describes what is happening in the system.
2. **Event sources** - An event source is where the event happens. Each event source is related to one or more event types.
3. **Topics** - The event grid topic provides an endpoint where the source sends events. A topic is used for a collection of related events.
4. **Event subscriptions** - A subscription tells Event Grid which events on a topic you're interested in receiving. You can filter the events that are sent to the endpoint.
5. **Event handlers** - An event handler is the place where the event is sent. The handler takes some further action to process the event.

Discover event schemas (1 of 2)

Event properties

Property	Type	Required
`topic`	string	No
`subject`	string	Yes
`eventType`	string	Yes
`eventTime`	string	Yes
`id`	string	Yes
`data`	object	No
`dataVersion`	string	No
`metadataVersion`	string	No

```
[
  {
    "topic": string,
    "subject": string,
    "id": string,
    "eventType": string,
    "eventTime": string,
    "data":{
      object-unique-to-each-publisher
    },
    "dataVersion": string,
    "metadataVersion": string
  }
]
```

Discover event schemas (2 of 2)

Azure Blob storage event example

```
[{
    "topic": "...", "subject": "...",
    "eventType": "Microsoft.Storage.BlobCreated",
    "eventTime": "2017-06-26T18:41:00.9584103Z",
    "id": "831e1650-001e-001b-66ab-eeb76e069631",
    "data": {
        "api": "PutBlockList", "eTag": "0x8D4BCC2E4835CD0",
        "storageDiagnostics": { "batchId": "b68529f3-68cd-4744-baa4-3c0498ec19f0" },
        "clientRequestId": "6d79dbfb-0e37-4fc4-981f-442c9ca65760",
        "requestId": "831e1650-001e-001b-66ab-eeb76e000000",
        "contentType": "application/octet-stream", "contentLength": 524288,
        "blobType": "BlockBlob", "sequencer": "00000000000004420000000000028963",
        "url": "https://test.blob.core.windows.net/container/blob"
    },
    "dataVersion": "", "metadataVersion": "1"
}]
```

Explore event delivery durability

- **Retry schedule** - When Event Grid receives an error for an event delivery attempt, it decides whether it should retry the delivery, dead-letter the event, or drop the event based on the type of the error.
- **Retry policy** - You can customize the retry policy when creating an event subscription through the *maximum number of attempts* and *event time-to-live* settings.
- **Output batching** - You can configure Event Grid to batch events for delivery for improved HTTP performance in high-throughput scenarios.
- **Delayed delivery** - If an endpoint experiences delivery failures, Event Grid will delay the delivery and retry of events to that endpoint.
- **Dead-letter events** - If an event can not be delivered within a certain time period, or a certain number of attempts, it can send the undelivered event to a storage account.

Control access to events (1 of 2)

Built-in roles

Azure Event Grid allows you to control the level of access given to different users to do various management operations such as list event subscriptions, create new ones, and generate keys. Event Grid uses Azure role-based access control (Azure RBAC).

Role	Description
Event Grid Subscription Reader	Read Event Grid event subscriptions.
Event Grid Subscription Contributor	Manage Event Grid event subscription operations.
Event Grid Contributor	Create and manage Event Grid resources.
Event Grid Data Sender	Send events to Event Grid topics.

Control access to events (2 of 2)

Permissions for event subscriptions

If you're using an event handler that isn't a WebHook (such as an event hub or queue storage), you need write access to that resource. This permissions check prevents an unauthorized user from sending events to your resource.

Topic Type	Description
System topics	Need permission to write a new event subscription at the scope of the resource publishing the event.
Custom topics	Need permission to write a new event subscription at the scope of the event grid topic.

Receive events by using webhooks

Three Azure services

- Azure Logic Apps with Event Grid Connector
- Azure Automation via webhook
- Azure Functions with Event Grid Trigger

Two ways to verify subscription

- **Synchronous handshake** - At the time of event subscription creation, Event Grid sends a subscription validation event to your endpoint.
- **Asynchronous handshake** - In certain cases, you can't return the ValidationCode in response synchronously.

Filter events

- Event type filtering
  - By default, all event types for the event source are sent to the endpoint.
  - You can decide to send only certain event types to your endpoint.
- Subject filtering
  - For simple filtering by subject, specify a starting or ending value for the subject.
  - You can filter the subject with `/blobServices/default/containers/testcontainer` to get all events for that container but not other containers in the storage account.
- Advanced filtering
  - To filter by values in the data fields and specify the comparison operator, use the advanced filtering option.

Exercise: Route custom events to web endpoint by using Azure CLI

In this exercise you learn how to use the Azure CLI to Event Grid resources, subscribe to a custom topic, and send an event to a custom topic.

Objectives

- Create a resource group
- Enable an Event Grid resource provider
- Create a custom topic
- Create a message endpoint
- Subscribe to a custom topic
- Send an event to your custom topic
- Clean up resources

Summary and knowledge check

In this module, you learned how to:

- Describe how Event Grid operates and how it connects to services and event handlers.
- Explain how Event Grid delivers events and how it handles errors.
- Implement authentication and authorization.
- Route custom events to web endpoint by using Azure CLI.

1. Which event schema properties requires a value?
2. Which Event Grid built-in role is appropriate for managing Event Grid resources?

Module 2: Explore Azure Event Hubs

Learning objectives

- Describe the benefits of using Event Hubs and how it captures streaming data.
- Explain how to process events.
- Perform common operations with the Event Hubs client library.

Introduction

- Azure Event Hubs is a big data streaming platform and event ingestion service.
- It can receive and process millions of events per second.
- Data sent to an event hub can be transformed and stored by using any real-time analytics provider or batching/storage adapters.

Discover Azure Event Hubs (1 of 2)

- Azure Event Hubs provides a unified streaming platform with time retention buffer, decoupling event producers from event consumers.
- It is a scalable event-processing service that ingests and processes large volumes of events and data, with low latency and high reliability.

Feature	Description
Fully managed PaaS	Event Hubs is a fully managed service with little configuration or management overhead
Real-time and batch processing	Event Hubs uses a partitioned consumer model, enabling multiple applications to process the stream concurrently and letting you control the speed of processing.
Scalable	Scaling options, like Auto-inflate, scale the number of throughput units to meet your usage needs.
Rich ecosystem	Event Hubs for Apache Kafka ecosystems enables Apache Kafka (1.0 and later) clients and applications to talk to Event Hubs

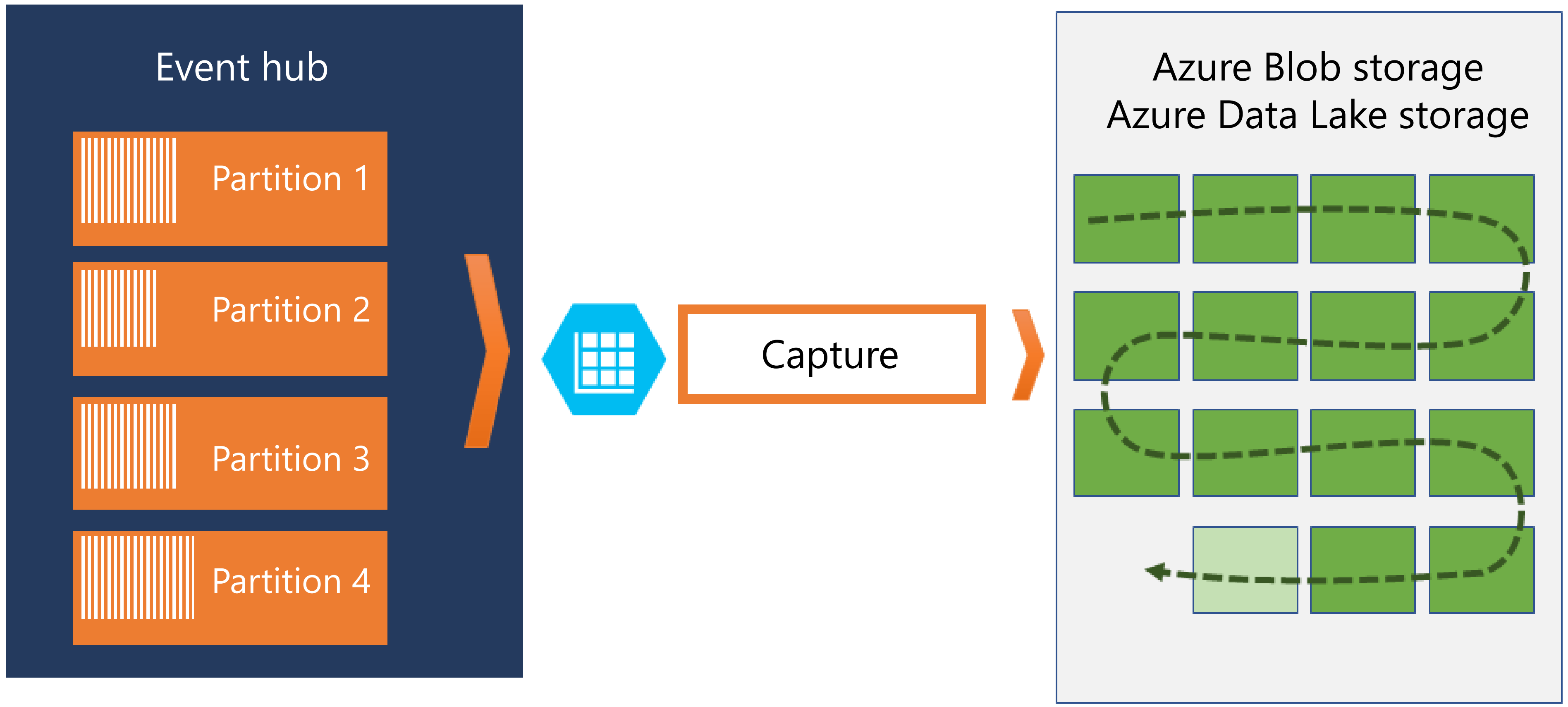
Discover Azure Event Hubs (2 of 2)

Key components

- An **Event Hub client** is the primary interface for developers interacting with the Event Hubs client library.
- An **Event Hub producer** is a type of client that serves as a source of telemetry data, diagnostics information, usage logs, or other log data, as part of an embedded device solution, a mobile device application, a game title running on a console or other device, some client or server based business solution, or a web site.
- An **Event Hub consumer** is a type of client which reads information from the Event Hub and allows processing of it. Processing may involve aggregation, complex computation and filtering.
- A **partition** is an ordered sequence of events that is held in an Event Hub. Partitions are a means of data organization associated with the parallelism required by event consumers.
- A **consumer group** is a view of an entire Event Hub. Consumer groups enable multiple consuming applications to each have a separate view of the event stream, and to read the stream independently at their own pace and from their own position.
- **Event receivers** - Any entity that reads event data from an event hub. All Event Hubs consumers connect via the AMQP 1.0 session.
- **Throughput units** or **processing units** - Pre-purchased units of capacity that control the throughput capacity of Event Hubs.

Explore Event Hubs Capture (1 of 2)

- Azure Event Hubs enables you to automatically capture the streaming data in Event Hubs in an Azure Blob storage or Azure Data Lake Storage account of your choice.
- Event Hubs Capture enables you to process real-time and batch-based pipelines on the same stream.



Explore Event Hubs Capture (2 of 2)

Capture windowing

- Event Hubs Capture enables you to set up a window to control capturing.
- Each partition captures independently and writes a completed block blob at the time of capture, named for the time at which the capture interval was encountered.

Scaling to throughput units

- Event Hubs traffic is controlled by throughput units.
- Event Hubs Capture copies data directly from the internal Event Hubs storage, bypassing throughput unit egress quotas and saving your egress for other processing readers.
- Event Hubs Capture runs automatically when you send your first event.

Scale your processing application (1 of 2)

The key to scale for Event Hubs is the idea of *partitioned consumers*. In contrast to the competing consumers pattern, the partitioned consumer pattern enables high scale by removing the contention bottleneck and facilitating end to end parallelism.

Example scenario

When designing the consumer in a distributed environment, the solution must handle the following requirements: scale, load balance, seamless resume on failures, and event consumption.

Event processor or consumer client

- You don't need to build your own solution to meet these requirements.
- The Azure Event Hubs SDKs provide this functionality.
- For most production scenarios use the event processor client for reading and processing events.

Scale your processing application (2 of 2)

Partition ownership tracking

- An event processor instance typically owns and processes events from one or more partitions.
- Each event processor is given a unique identifier and claims ownership of partitions by adding or updating an entry in a checkpoint store.

Receive messages

- When you create an event processor, you specify the functions that will process events and errors.
- We recommend that you do things relatively fast. That is, do as little processing as possible.

Checkpointing

- Checkpointing is a process by which an event processor marks or commits the position of the last successfully processed event within a partition.
- By specifying a lower offset from this checkpoint process, you can return to older data.

Thread safety and processor instances

By default, the function that processes the events is called sequentially for a given partition. As the event pump continues to run in the background of other threads, subsequent events from the same partition and calls to this function are queued in the background.

Control access to events

Azure built-in roles:

- Azure Event Hubs Data Owner: Use this role to give *complete access* to Event Hubs resources.
- Azure Event Hubs Data Sender: Use this role to give *send access* to Event Hubs resources.
- Azure Event Hubs Data Receiver: Use this role to give *receiving access* to Event Hubs resources.

You can:

- Authorize access with managed identities
- Authorize access with Microsoft Identity Platform
- Authorize access to Event Hubs publishers with shared access signatures
- Authorize access to Event Hubs consumers with shared access signatures

Perform common operations with the Event Hubs client library (1 of 3)

Inspect an Event Hub

```
await using (var producer = new EventHubProducerClient(connectionString, eventHubName))
{
    string[] partitionIds = await producer.GetPartitionIdsAsync();
}
```

Perform common operations with the Event Hubs client library (2 of 3)

Publish events to an Event Hub

```
await using (var producer = new EventHubProducerClient(connectionString, eventHubName))
{
    using EventDataBatch eventBatch = await producer.CreateBatchAsync();
    eventBatch.TryAdd(new EventData(new BinaryData("First")));
    eventBatch.TryAdd(new EventData(new BinaryData("Second")));

    await producer.SendAsync(eventBatch);
}
```

Perform common operations with the Event Hubs client library (3 of 3)

Read events from an Event Hub

```
string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;

await using (var consumer = new EventHubConsumerClient(consumerGroup, connectionString, eventHubName))
{
    using var cancellationSource = new CancellationTokenSource();
    cancellationSource.CancelAfter(TimeSpan.FromSeconds(45));

    await foreach (PartitionEvent receivedEvent in consumer.ReadEventsAsync(cancellationSource.Token))
    {
        // At this point, the loop will wait for events to be available in the Event Hub.  When an event
        // is available, the loop will iterate with the event that was received.  Because we did not
        // specify a maximum wait time, the loop will wait forever unless cancellation is requested using
        // the cancellation token.
    }
}
```

Summary and knowledge check

In this module, you learned how to:

- Describe the benefits of using Event Hubs and how it captures streaming data.
- Explain how to process events.
- Perform common operations with the Event Hubs client library.

1. Which Event Hubs concept represents an ordered sequence of events that is held in an Event Hubs?
2. Which process represents when an event processor marks or commits the position of the last successfully processed event within a partition?

Discussion and lab

Group discussion questions

- Can you describe the differences between the capabilities of Azure Event Grid and Azure Event Hubs? When would you use one over the other?
- Contoso Inc is using Event Grid in their solution, but many events are being lost during delivery. What can they do to improve delivery and retain the events being lost?
- Can you list the three Azure built-in roles for Event Hubs and what permissions they grant?

Lab 09: Publish and subscribe to Event Grid events

In this lab, you will start with a proof-of-concept web app, hosted in a container, that will be used to subscribe to your Event Grid. This app will allow you to submit events and receive confirmation messages that the events were successful.

http://aka.ms/az204labs

- Exercise 1: Create Azure resources
- Exercise 2: Create an Event Grid subscription
- Exercise 3: Publish Event Grid events from .NET

End of presentation