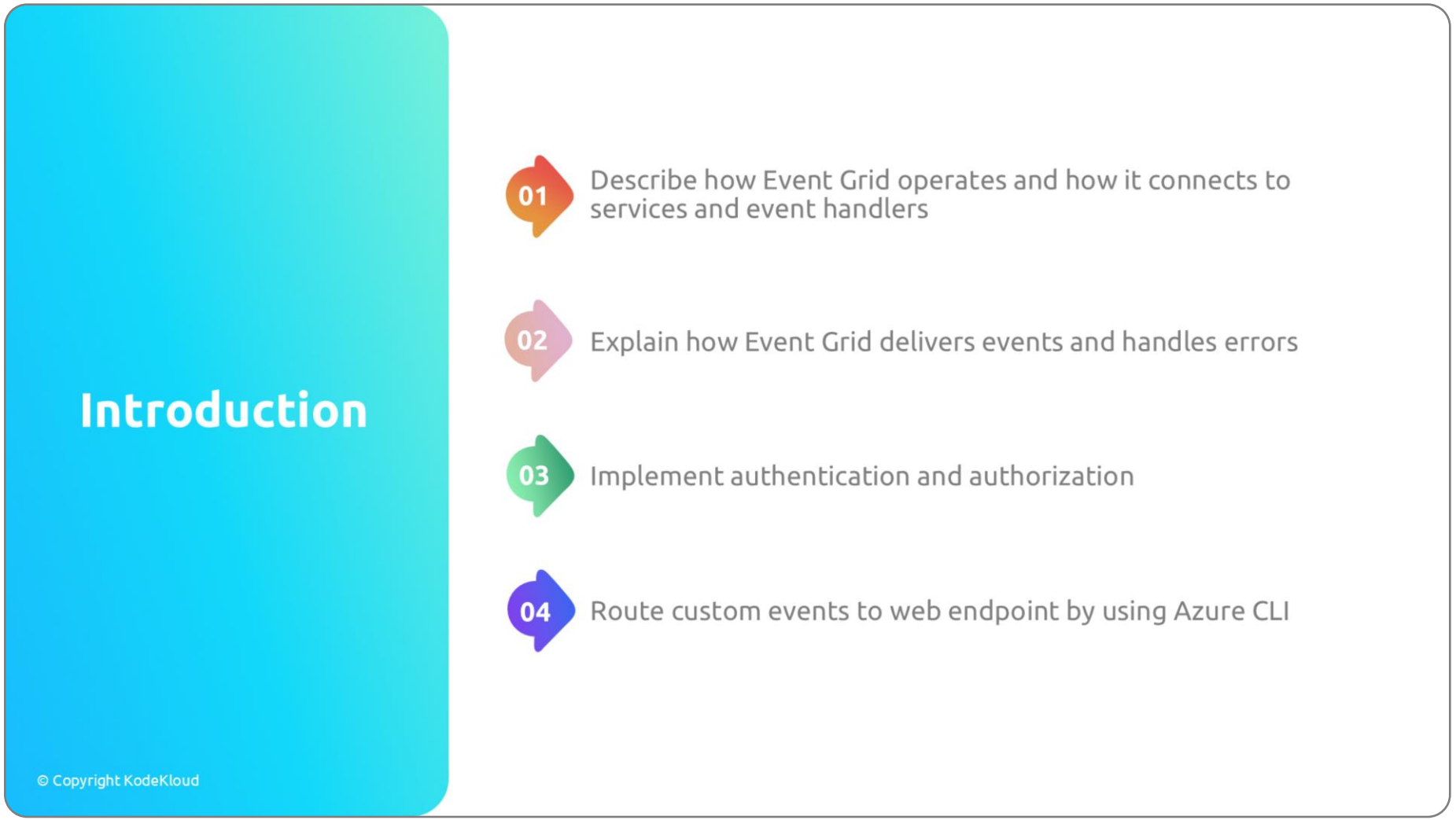
KodeKloud

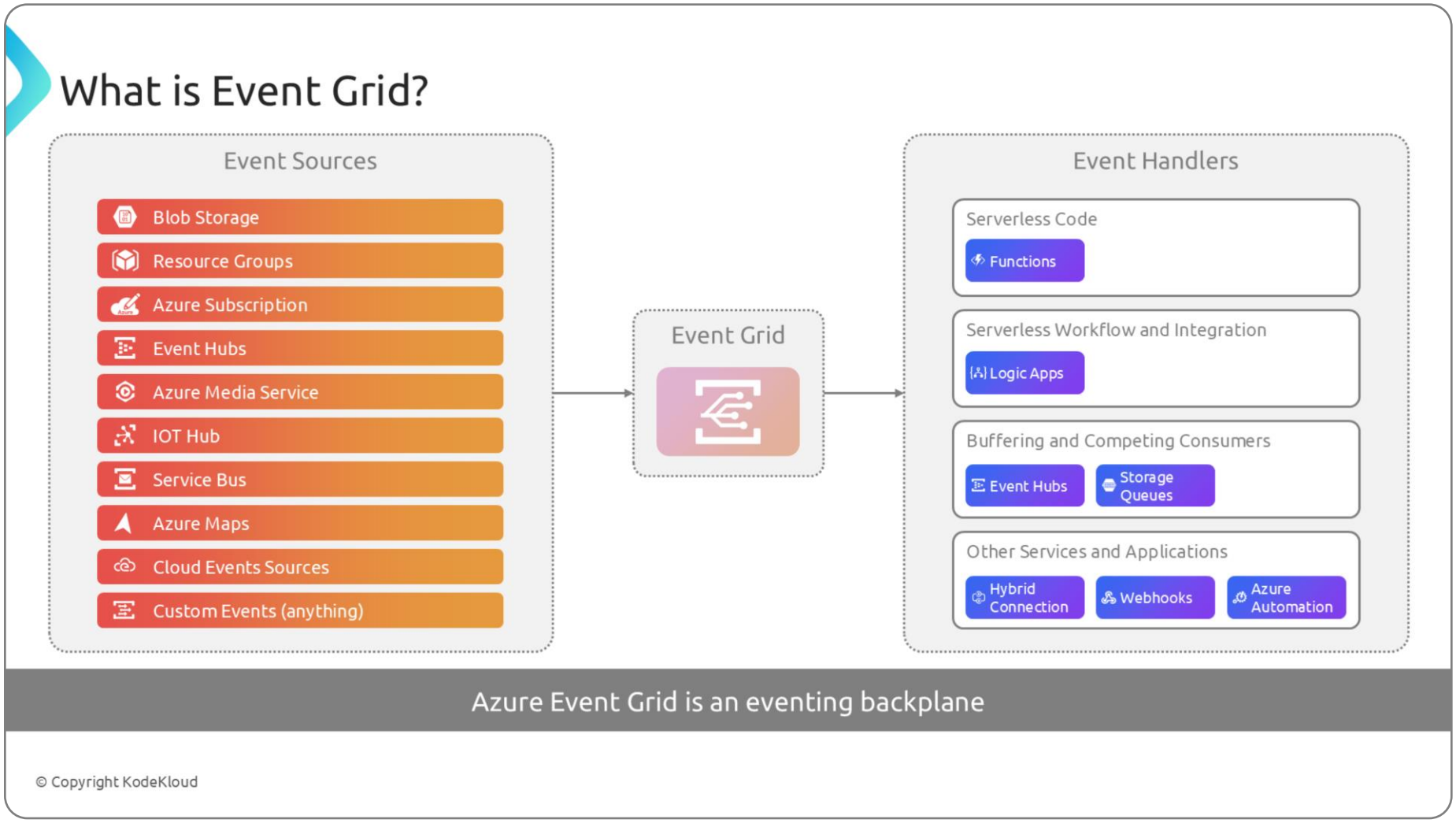
Visit www.kodekloud.com to learn more.

Exploring Azure Event Grid



Module overview

Azure Event Grid – Overview



In an event-driven architecture, events are delivered in nearly real time, so that consumers can respond immediately to events as they occur.

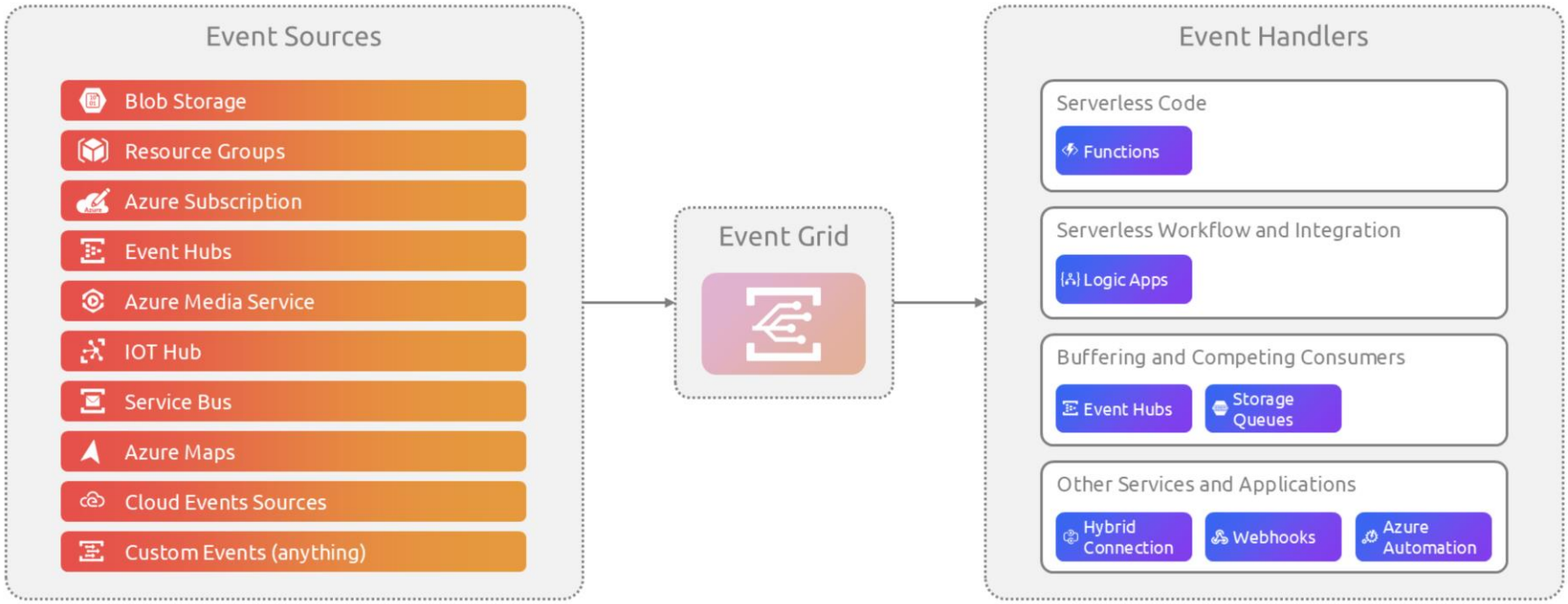
An event-driven architecture can use a pub/sub model or an event-stream model.
Pub/sub: The messaging infrastructure keeps track of subscriptions. When an event is published, it sends the event to each subscriber. After an event is received, it cannot be replayed, and new subscribers do not observe the event.
Event streaming: Events are written to a log. Events are strictly ordered (within a partition) and durable. Clients don't

subscribe to the stream. Instead, a client can read from any part of the stream. The client is responsible for advancing its position in the stream. That means that a client can join at any time, and can replay events.

On the consumer side, there are some common variations:
Simple event processing. An event immediately triggers an action in the consumer. For example, you could use Microsoft Azure Functions with a Service Bus trigger, so that a function executes whenever a message is published to a Service Bus topic.
Complex event processing. A consumer processes a series of events, observing patterns in the event data by using a technology such as Azure Stream Analytics or Apache Storm. For example, you could aggregate readings from an embedded device over a time window, and generate a notification if the moving average crosses a certain threshold.
Event stream processing. Use a data-streaming platform, such as Azure IoT Hub or Apache Kafka, as a pipeline to ingest events and feed them to stream processors. The stream processors act to process or transform the stream. There may be multiple stream processors for different subsystems of the application. This approach is a good fit for Internet of Things (IoT)workloads.

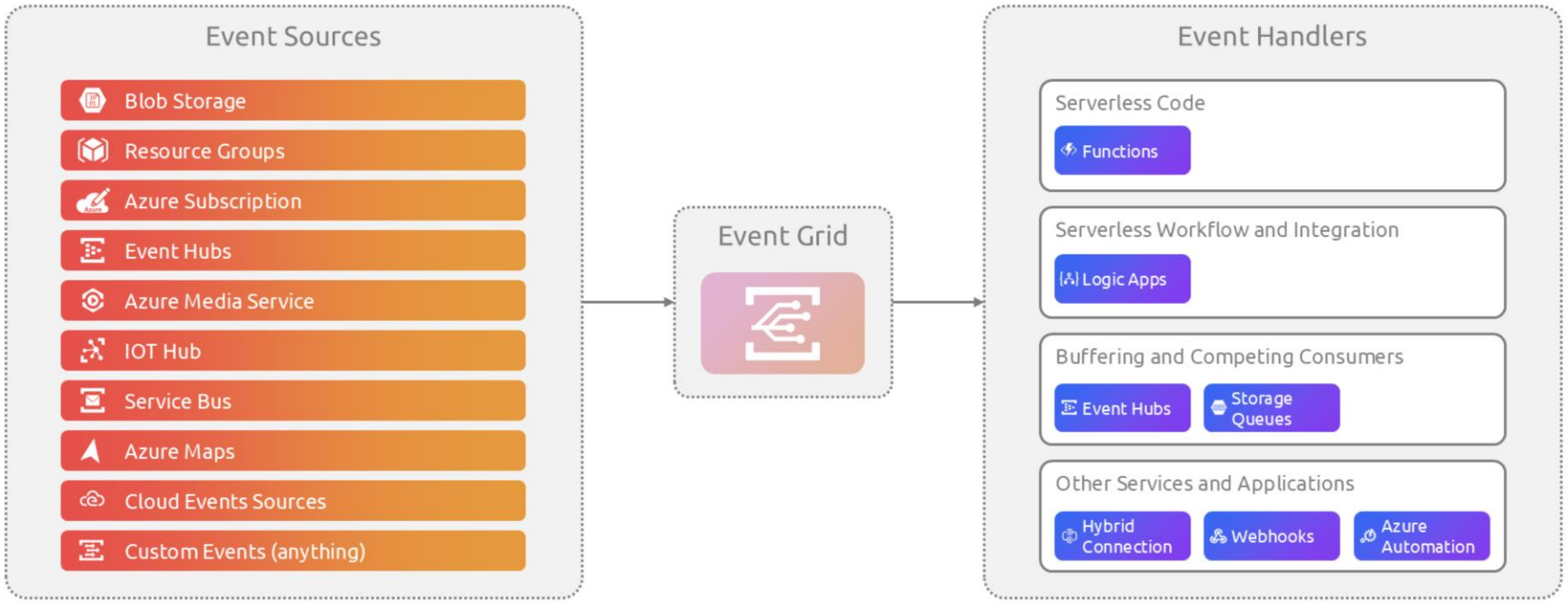
The source of the events might be external to the system, such as physical devices in an IoT solution. In that case, the system must be able to ingest the data at the volume and throughput that is required by the data source.

What is Event Grid?



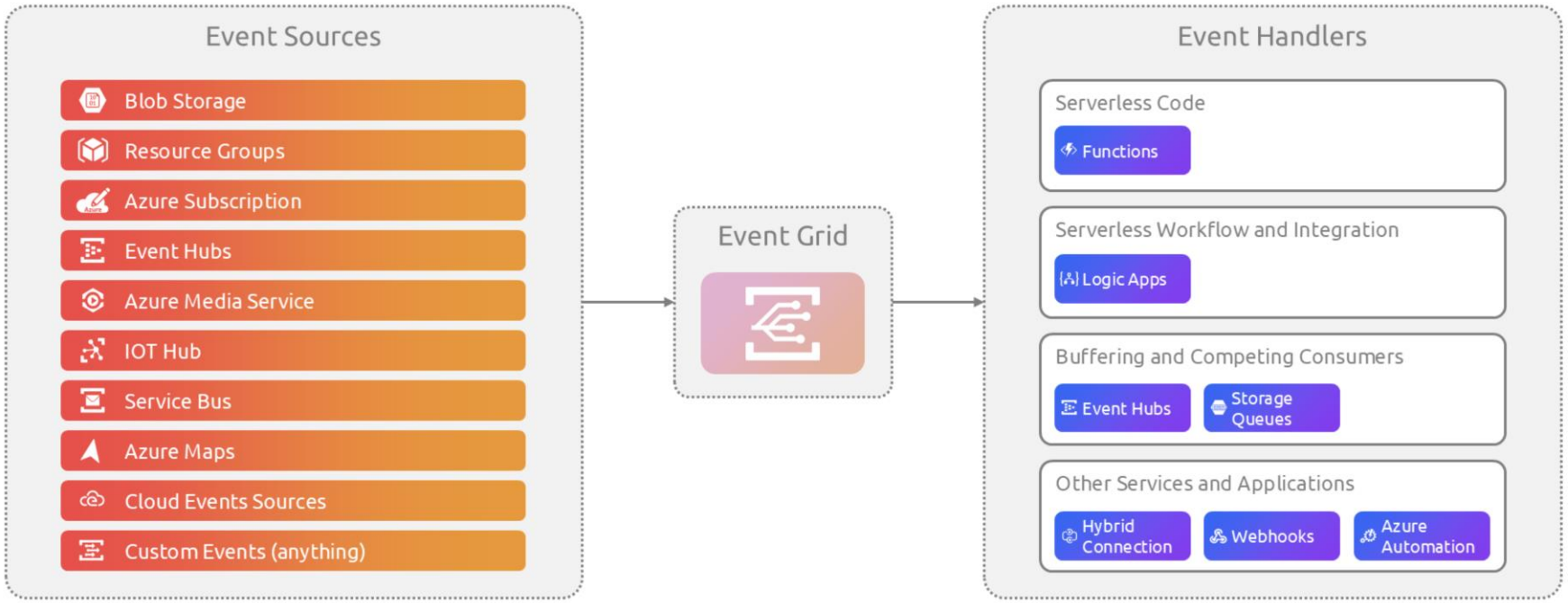
Built-in support for events comes from Azure services

What is Event Grid?



Create events with custom topics

What is Event Grid?



Use filters to route specific events to different endpoints

Five Key Concepts

Events

- Smallest unit of information describing system activity
- Provides comprehensive system activity details

Event Sources

- Location where events occur
- Each event source is associated with one or more event types

Topics

- Endpoint for event transmission from the source
- Topic groups-related events for efficient management

Event Subscriptions

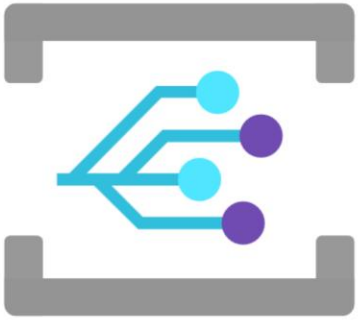
- Informs Event Grid about the desired events on a topic
- Allows filtering of events sent to the specified endpoint

Event Handlers

- Destination for the event transmission
- Executes additional actions to process the received event

Discovering Event Schemas

Discovering Event Schemas



Structure of events → Azure Event Grid

Event properties → Data transmission

Discovering event schemas So in this module we'll be focusing on the structure of events that are processed through Azure Event Grid and using the event properties to understand the data being transmitted So the event schema defines the structure of an event in Azure event grid

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

Example

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


Property descriptions
The example shows the properties that are used by all event publishers

topic: Full resource path to the event source. If not included, Event Grid will stamp onto the event. If included it must match the event grid topic Azure Resource Manager ID exactly. Event Grid provides this value.
subject: Publisher-defined path to the event subject.
eventType: One of the registered event types for this event source.

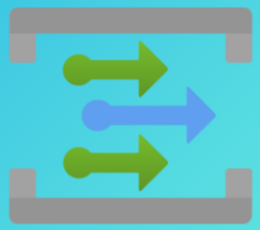
eventTime: The time the event is generated based on the provider's UTC time.
id: Unique identifier for the event.
data: Event data specific to the resource provider.
dataVersion: The schema version of the data object. The publisher defines the schema version.
metadataVersion: The schema version of the event metadata. Event Grid defines the schema of the top-level properties.

Azure Blob Storage Event – Example

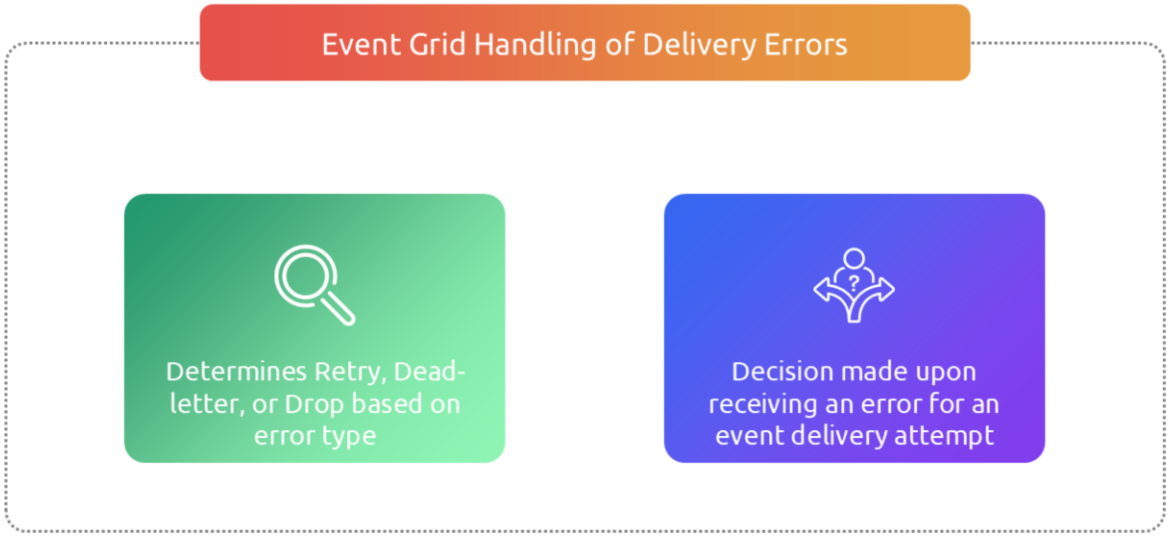
Example

```
[{
    "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"
}]
```

Exploring Event Delivery Durability



Retry Schedule



Event Grid provides durable delivery. It tries to deliver each event at least once for each matching subscription immediately. If a subscriber's endpoint doesn't acknowledge receipt of an event or if there is a failure, Event Grid retries delivery based on a fixed retry schedule and retry policy.
Event Grid doesn't guarantee order for event delivery, so subscribers may receive them out of order.

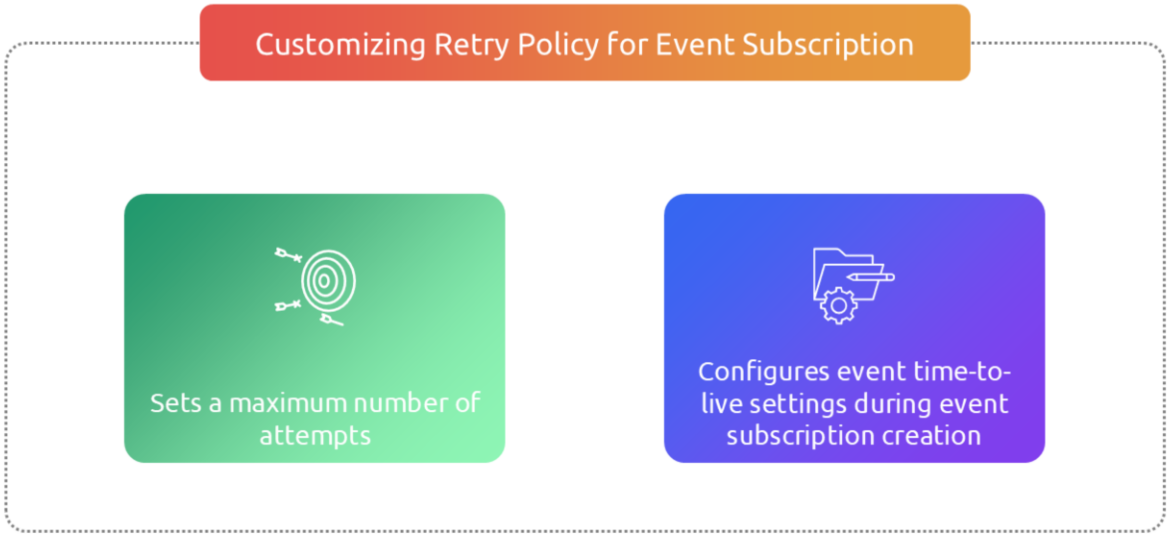
Retry schedule
If the error returned by the subscribed endpoint is a configuration-related error that can't be fixed with retries (for

example, if the endpoint is deleted), EventGrid will either perform dead-lettering on the event or drop the event if dead-letter isn't configured.
If Dead-Letter isn't configured for an endpoint, events will be dropped when those types of errors happen. Consider configuring Dead-Letter if you don't want these kinds of events to be dropped.

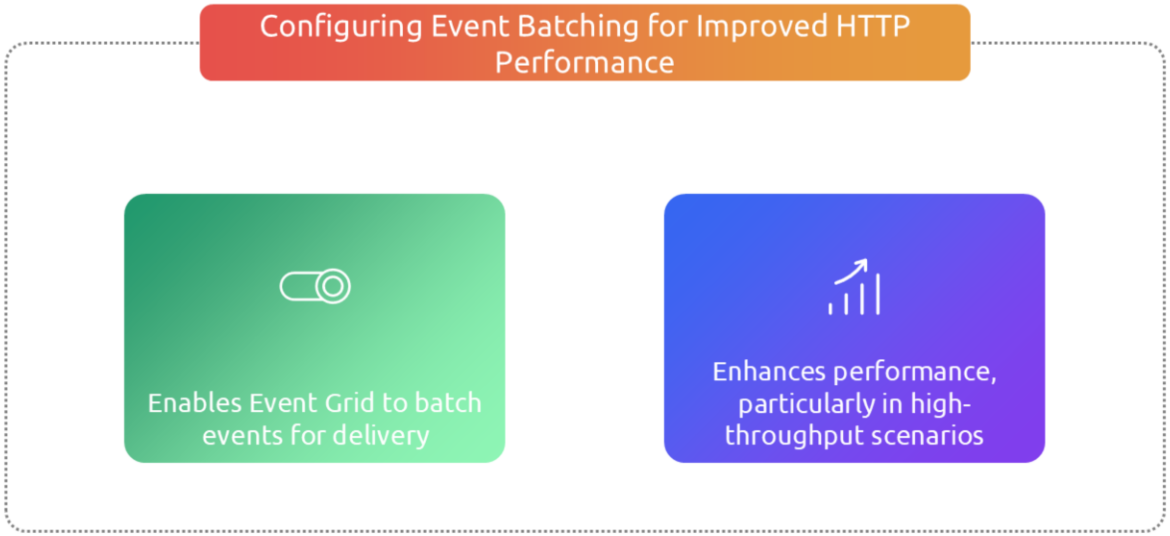
Output batching
Batching is turned off by default and can be turned on per-subscription via the portal, CLI, PowerShell, or SDKs.
Two settings:

- Max events per batch - Maximum number of events Event Grid will deliver per batch. This number will never be exceeded, however fewer events may be delivered if no other events are available at the time of publish.
- Preferred batch size in kilobytes -

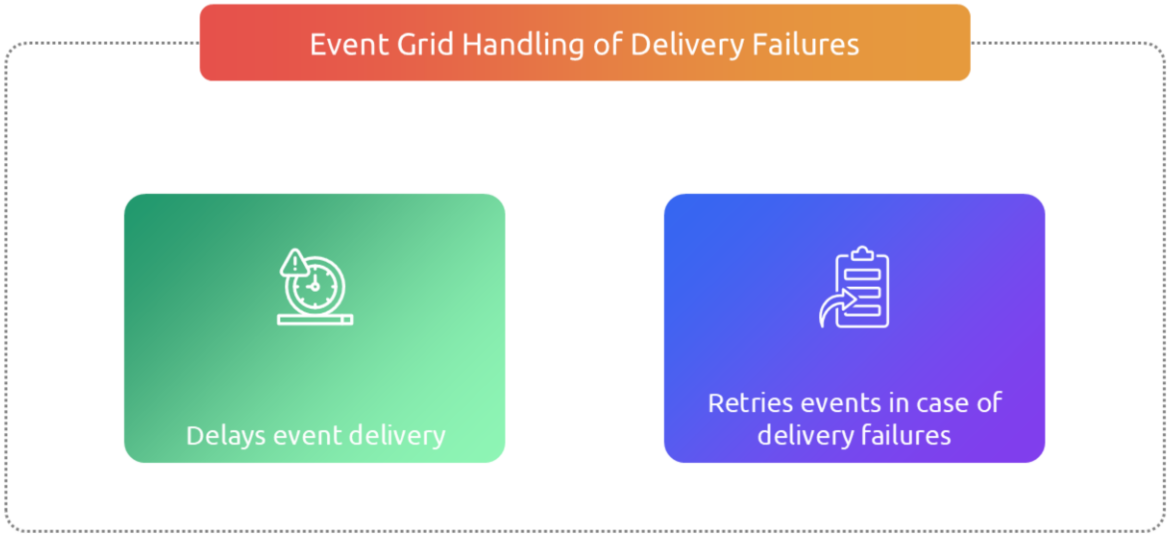
Retry Policy



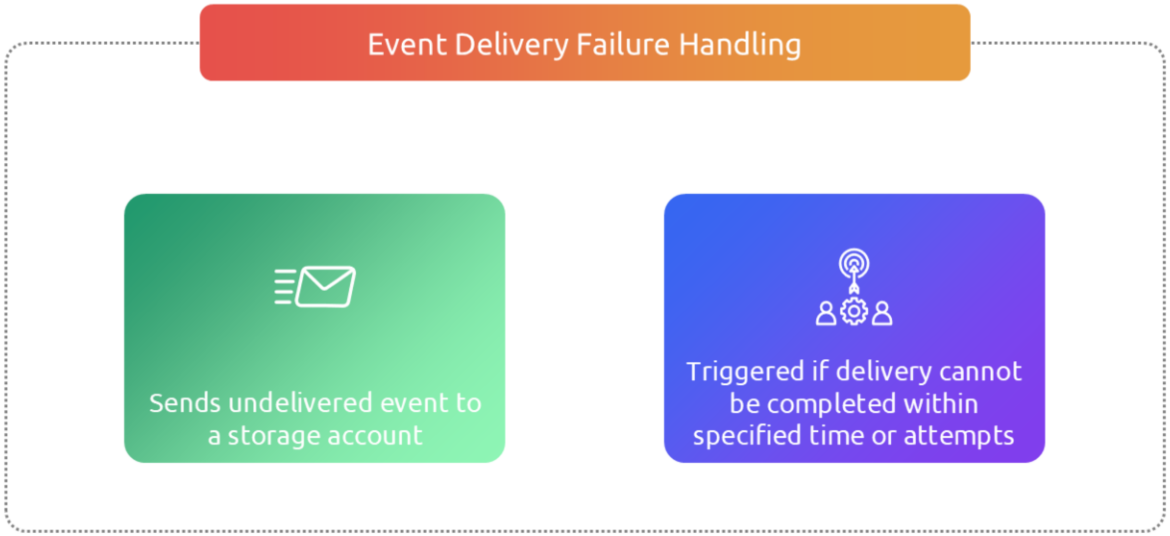
Output Batching



Delayed Delivery

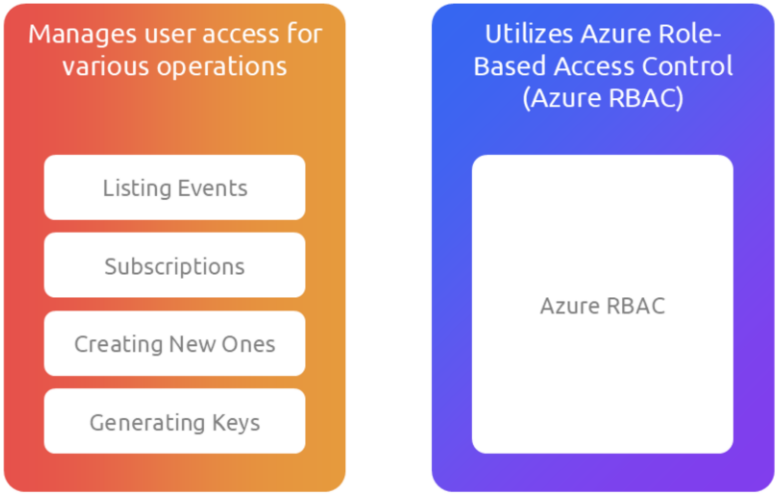


Dead-Letter Events



Controlling Access to Events

Azure Event Grid Access Control



Built-in Roles

Role	Description
Event Grid Subscription Reader	Lets you read Event Grid event subscriptions
Event Grid Subscription Contributor	Lets you manage Event Grid event subscription operations
Event Grid Contributor	Lets you create and manage Event Grid resources
Event Grid Data Sender	Lets you send events to Event Grid topics

Permissions for Event Subscriptions

For non-WebHook event handlers (e.g., event hub, queue storage), you require write access to the resource

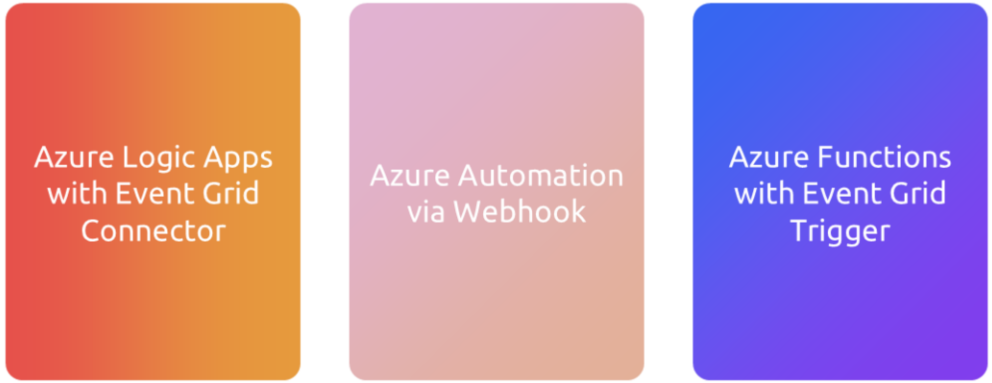
Permissions check ensures only authorized users can send events to the resource

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

You must have the **Microsoft.EventGrid/EventSubscriptions/Write** permission on the resource that is the event source. You need this permission because you're writing a new subscription at the scope of the resource. The required resource differs based on whether you're subscribing to a system topic or custom topic.

Receiving Events by Using Webhooks

Three Azure Services



Two Ways to Verify Subscription

Synchronous Handshake

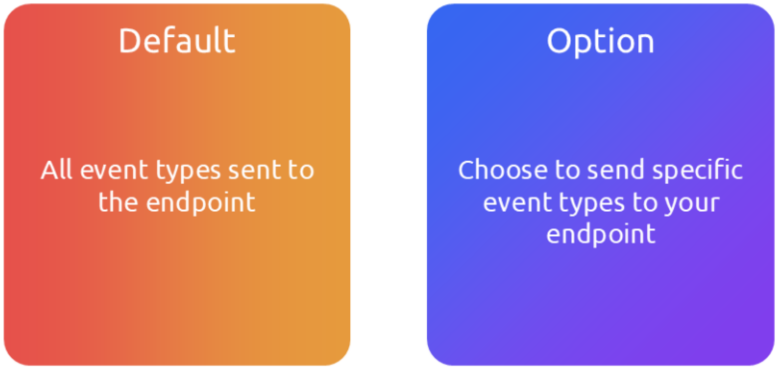
At the time of event subscription creation, Event Grid sends a subscription validation event to your endpoint.

Asynchronous Handshake

In certain cases, you can't return the Validation Code in response synchronously.

Filtering Events

Event Type Filtering



Subject Filtering

Specify starting or ending value for the subject

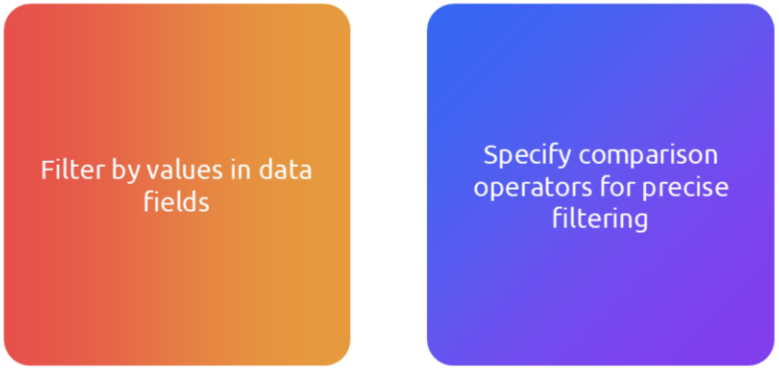
Example

Filter subject begins with

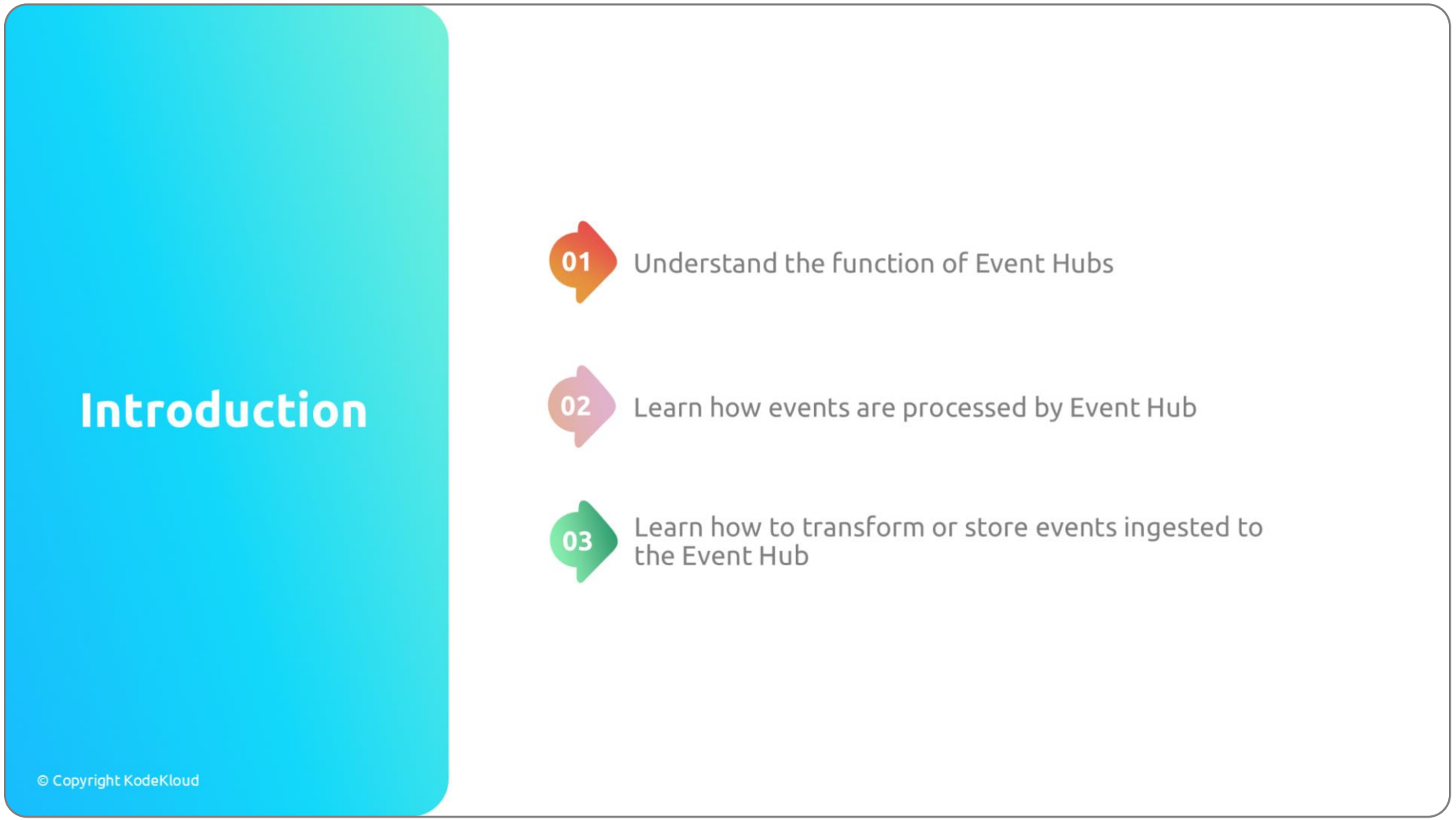
"/blobServices/default/containers/testcontainer"

for events specific to that container

Advanced Filtering



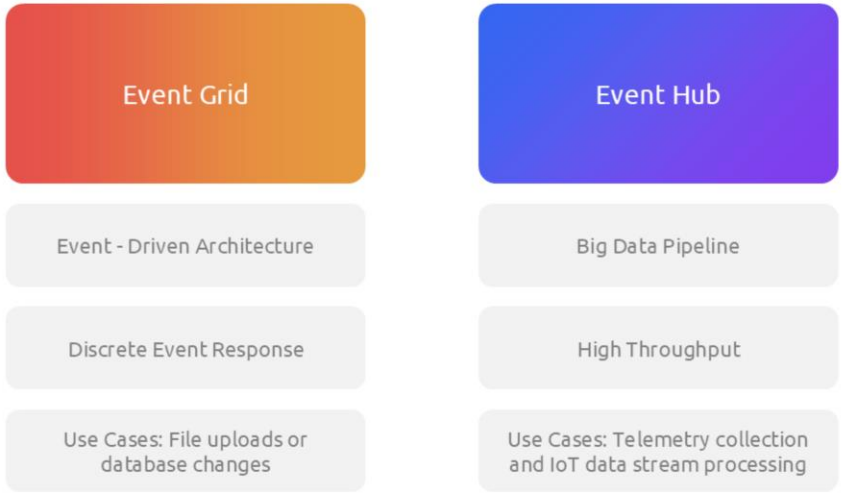
Exploring Azure Event Hubs



Module overview

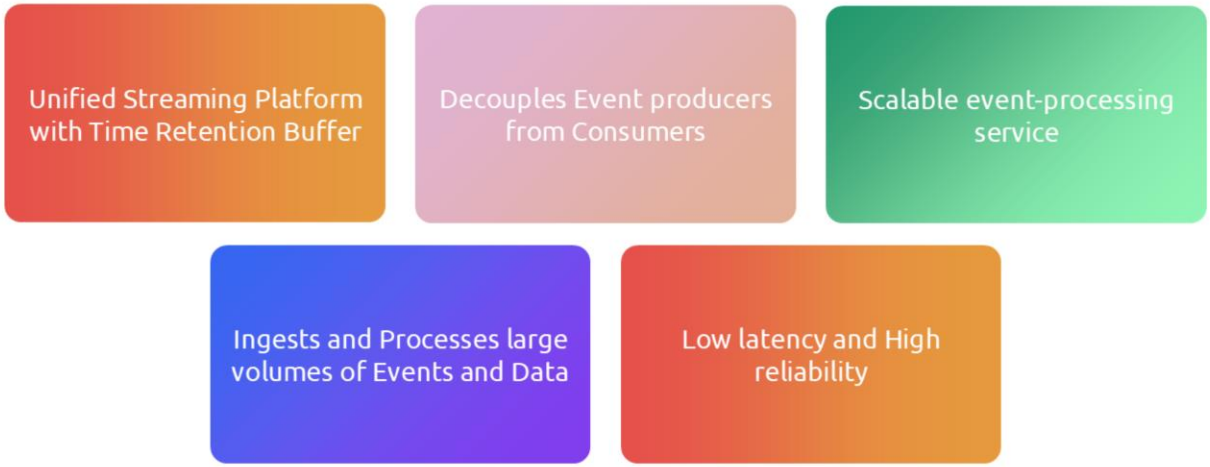
Discovering Azure Event Hubs

Azure Event Hubs – Overview



Discovering Azure even hubs To begin this section, let's compare even grid and even hub to help you understand when to use each service So even grid is best used for event driven architecture where the focus is on the lightweight even notification It's ideal for scenarios where services need to respond to discrete changes or events such as file uploads to a storage account or changes to a database, et cetera Even hubs on the other hand, is a big data pipeline for scenarios where you need to ingest and process large volumes of streaming data It is designed for high throughput and supports realtime analytics, making it perfect for telemetry collection or processing data streams from IOT devices

Azure Event Hubs – Overview



Features

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 enable Apache Kafka (1.0 and later) clients and applications to talk to Event Hubs.

Azure Event Hubs 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

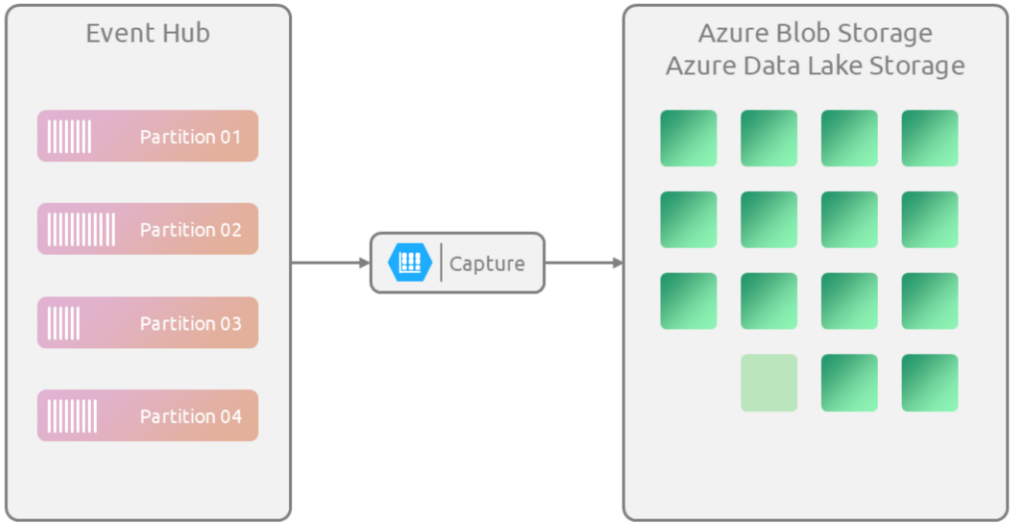
associated with the parallelism required by event consumers.
A consumer group is a view of an entire Event Hub. Consumer groups enable multiple consuming applications to each have a separate view of the event stream, and to read the stream independently at their own pace and from their own position.
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.

Exploring Event Hubs Capture

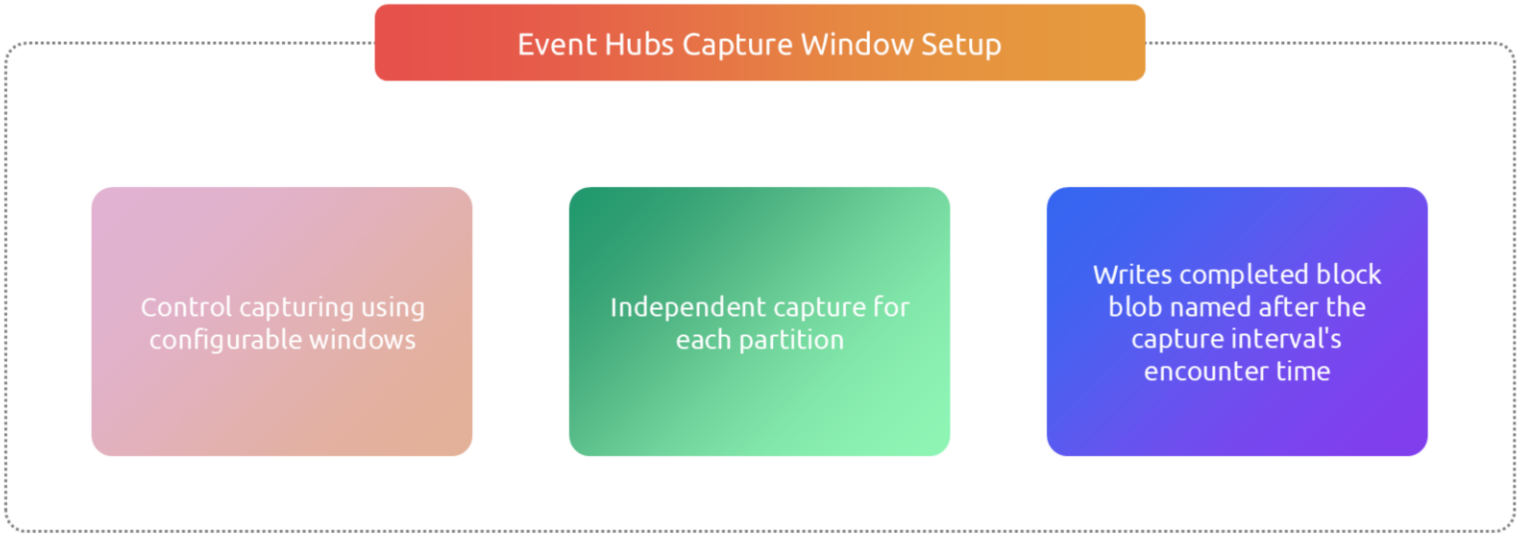
Azure Event Hubs Capture – Features

Automatic streaming data capture in Azure Blob or Data Lake Storage

Enables real-time and batch-based pipelines on the same Stream

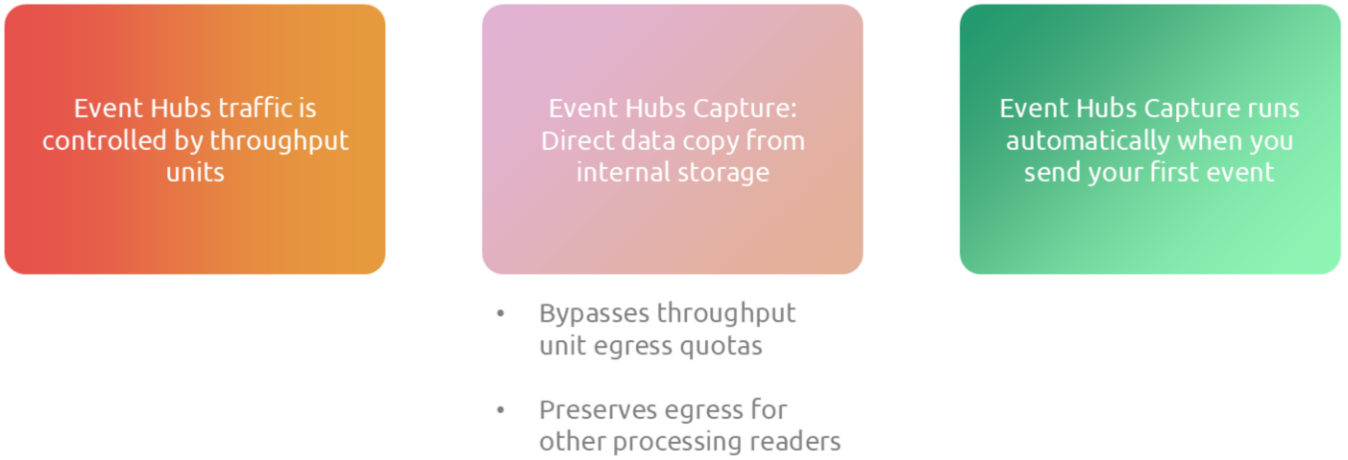


Capture Windowing



Capture windowing
This window is a minimum size and time configuration with a "first wins policy," meaning that the first trigger encountered causes a capture operation.

Scaling to Throughput Units



Scaling to throughput units
A single throughput unit allows 1 MB per second or 1000 events per second of ingress and twice that amount of egress. Standard Event Hubs can be configured with 1-20 throughput units, and you can purchase more with a quota increase support request.

Scaling Your Processing Application

Scaling With Partitioned Consumers in Event Hubs

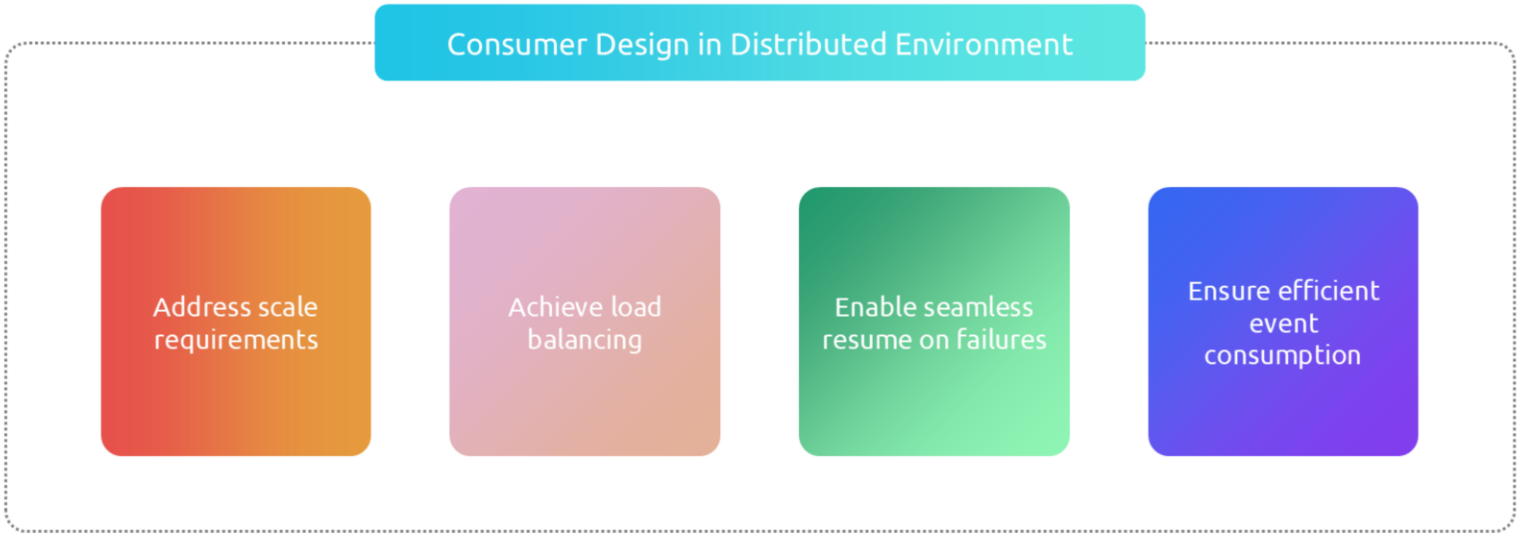
Key to Scale: Partitioned Consumers

Partitioned Consumer pattern enables high scale

Removes contention bottleneck

Facilitates end-to-end parallelism

Example Scenario



Event Processor or Consumer Client

No need to build a custom solution

Azure Event Hubs SDKs offer required functionality

Utilize event processor client for event reading and processing in production

Partition Ownership Tracking

Typically processes events from one or more partitions

Assigned a unique identifier

Claims partition ownership through checkpoint store entries

Receiving Messages

Typically processes events from one or more partitions

Recommended: Optimize for speed, minimizing processing tasks

Checkpointing

Marks or commits position of last successfully processed event in a partition

Specifying a lower offset from this checkpoint allows revisiting older data

Thread Safety and Processor Instances

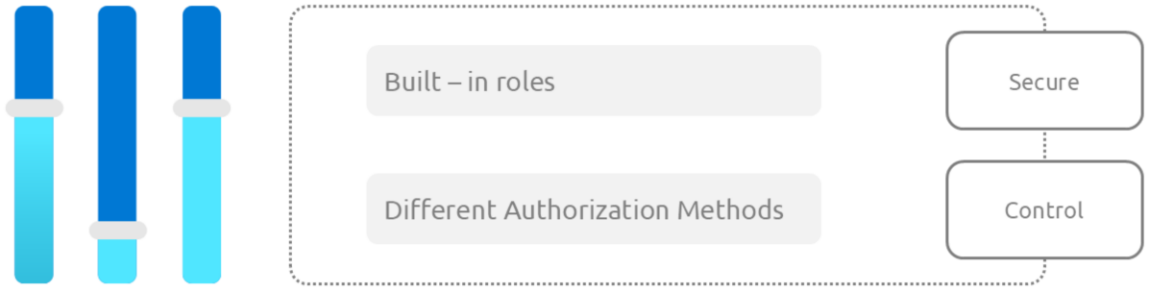
Events processed sequentially for a given partition

Event pump operates in the background with other threads

Subsequent events and function calls are queued for background processing

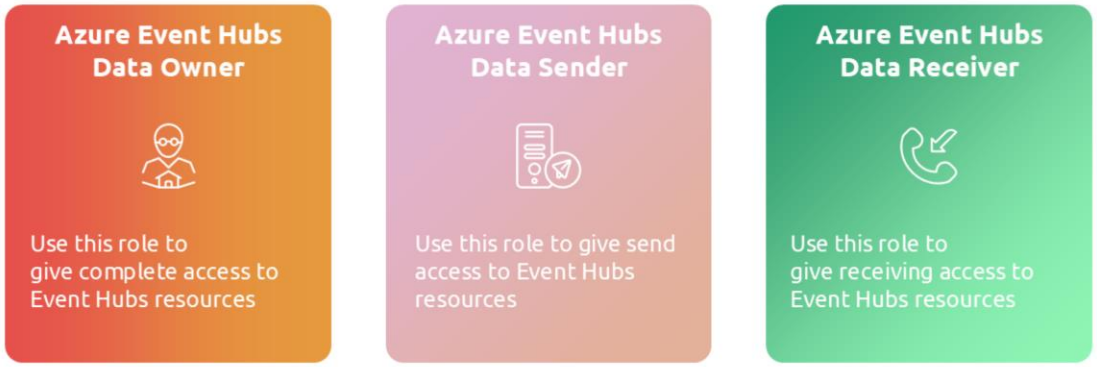
Controlling Access to Events

Controlling Access to events



Controlling access to events In this section, we are looking at how Azure controls access to events in event hubs, so builtin roles and different authorization methods These roles and access configurations help secure your data and control the flow of events between different entities

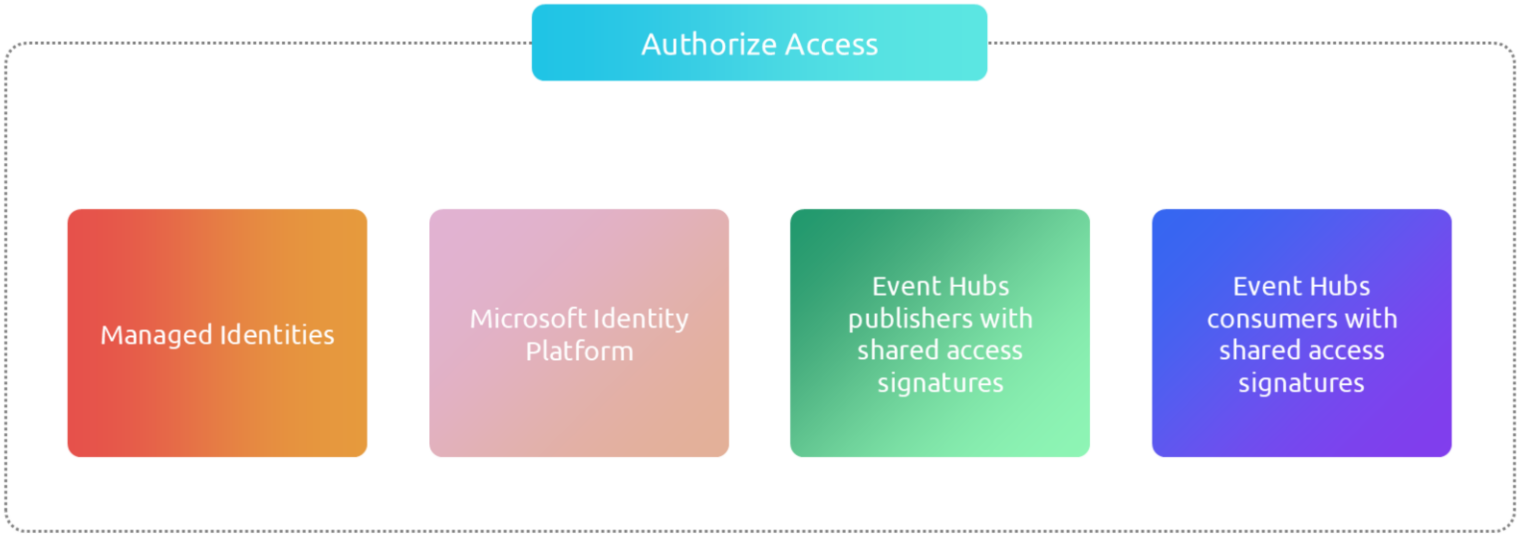
Azure Built-in Roles



Authorize access with managed identities
To authorize a request to Event Hubs service from a managed identity in your application, you need to configure Azure role-based access control settings for that managed identity.
Authorize access with Microsoft Identity Platform
A key advantage of using Azure AD with Event Hubs is that your credentials no longer need to be stored in your code. Instead, you can request an OAuth 2.0 access token from Microsoft identity platform.
Authorize access to Event Hubs publishers with shared access signatures

An event publisher defines a virtual endpoint for an Event Hub. The publisher can only be used to send messages to an event hub and not receive messages.
Each Event Hubs client is assigned a unique token which is uploaded to the client.
All tokens are assigned with shared access signature keys.
Authorize access to Event Hubs consumers with shared access signatures
To authenticate back-end applications that consume from the data generated by Event Hubs producers, Event Hubs token authentication requires its clients to either have the manage rights or the listen privileges assigned to its Event Hubs namespace or event hub instance or topic.

You Can...



Performing Common Operations With the Event Hubs Client Library

Inspect an Event Hub

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


Many Event Hub operations take place within the scope of a specific partition.
To understand what partitions are available, you query the Event Hub using one of the Event Hub clients.
The EventHubProducerClient is demonstrated in these examples, but the concept and form are common across clients.

Performing Common Operations With the Event Hubs Client Library

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);
}
```


Producers publish events in batches and may request a specific partition, or allow the Event Hubs service to decide which partition events should be published to.
It is recommended using automatic routing when the publishing of events needs to be highly available or when event data should be distributed evenly among the partitions.

Performing Common Operations With the Event Hubs Client Library

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.
    }
}
```


Our example reads all events that have been published to the Event Hub using an iterator.
This approach to consuming is intended to provide a general example.
For production use it is recommended to use the Event Processor Client (https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/eventhub/Azure.Messaging.EventHubs.Processor), as it provides a more robust and performant experience.

KodeKloud

Visit www.kodekloud.com to learn more.