avanade

Do what matters

Distributed, Event-Driven systems deep-dive

Event Hub deep dive

Session 2 (1 hour). Deep dive on Azure Event Hub.

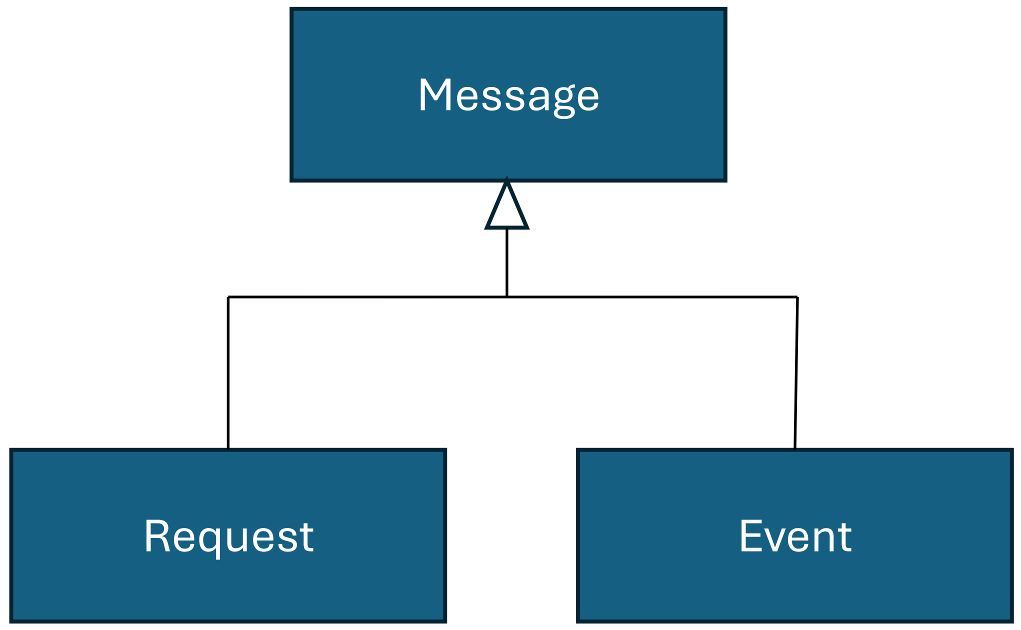
Again, some theory and a demo will be provided, as will access to a repo with the code used.

Agenda

- Some Theory
- About Azure Event Hub
- Azure SDK Event Hub clients
- PoC description
- PoC: Event Streams – demonstration, code
- PoC: Function triggered by Event Hub – demonstration, code
- Lessons learned

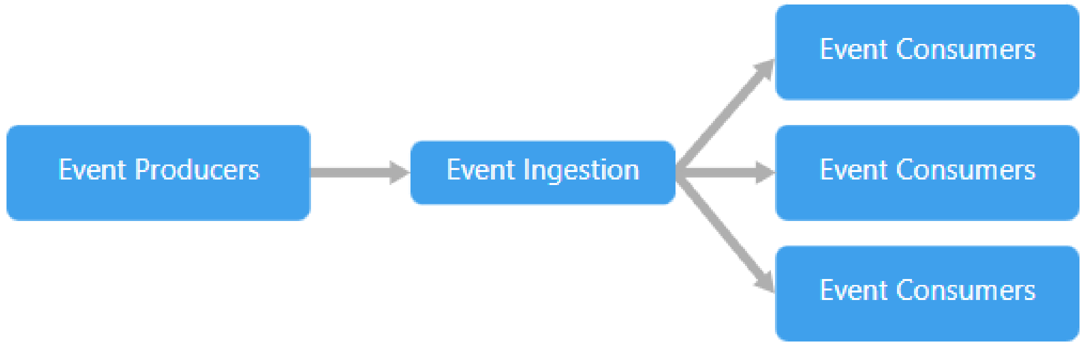
What is an Event?

- A *Message* is a payload of data sent from one system to another.
- A *Request* is a type of message that asks for an action to be performed (in the future).
  - E.g., “Please process the data”
- An *Event* is a type of message that notifies something has happened (in the past).
  - E.g., “The processing is complete”



Event-Driven Architecture

- An event-driven architecture consists of **event producers** that generate a stream of events, and **event consumers** that listen for the events.
- Events are delivered in near real time, so consumers *can* respond immediately to events as they occur.
- Producers are decoupled from consumers — a producer doesn't know which consumers are listening.
- Consumers are also decoupled from each other, and every consumer sees all of the events.



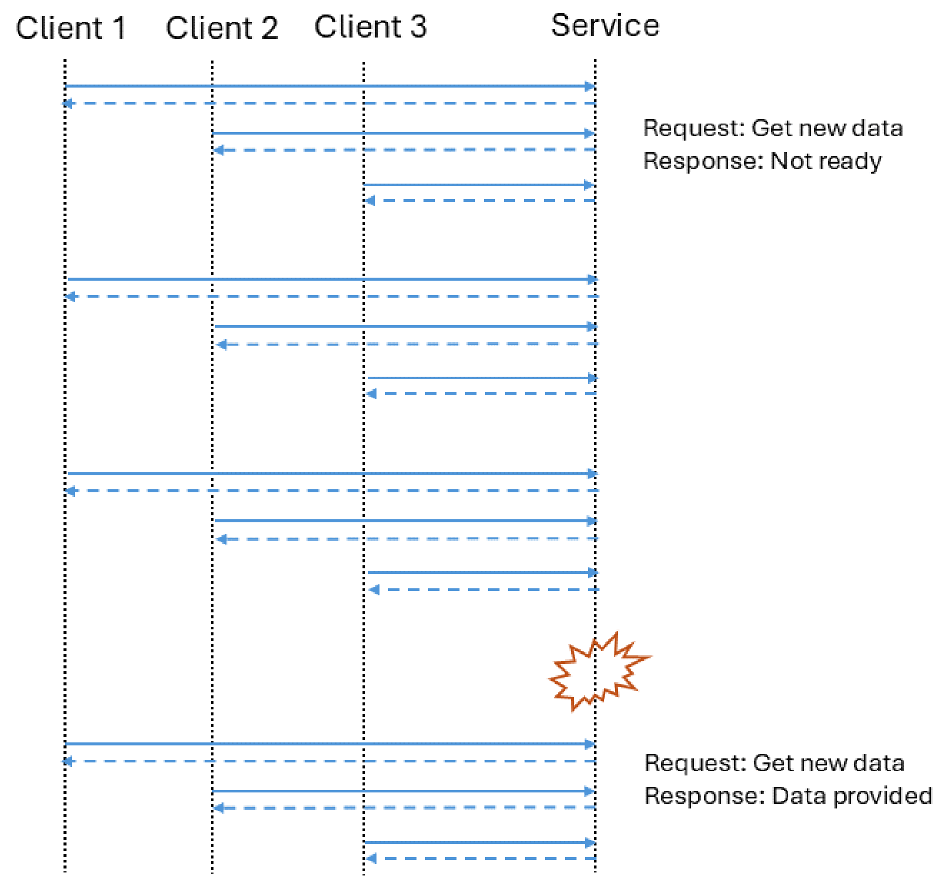
[Event-driven architecture style - Azure Architecture Center | Microsoft Learn]

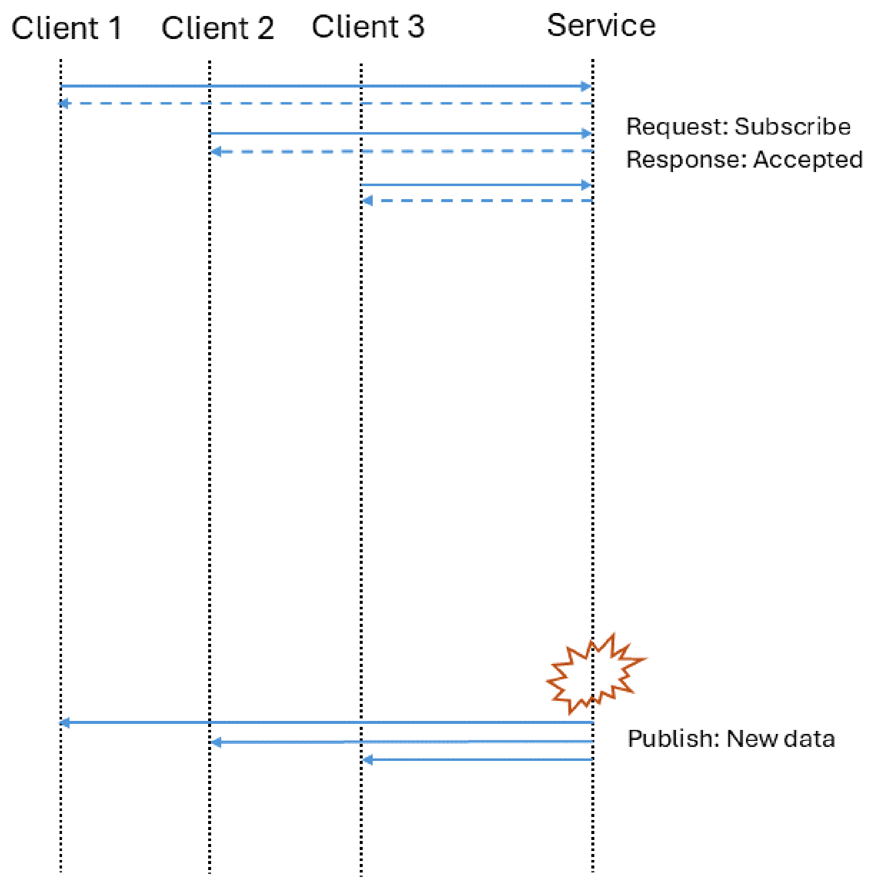
Publish-Subscribe vs. Event Streaming

Pub-Sub	Event Streaming
The messaging infrastructure *(Event Ingestion)* keeps track of subscriptions.	Events are written to a log. Events are strictly ordered (within a partition) and durable.
When an event is published, it sends the event to each subscriber *(Consumer).*	Clients (*Consumers*) don't subscribe to the stream, instead a client can read from any part of the stream.
After an event is received, it can't be replayed, and new subscribers don't see the event	The client is responsible for advancing its position in the stream. That means a client can join at any time, and can replay events.

Event-driven architecture style - Azure Architecture Center | Microsoft Learn

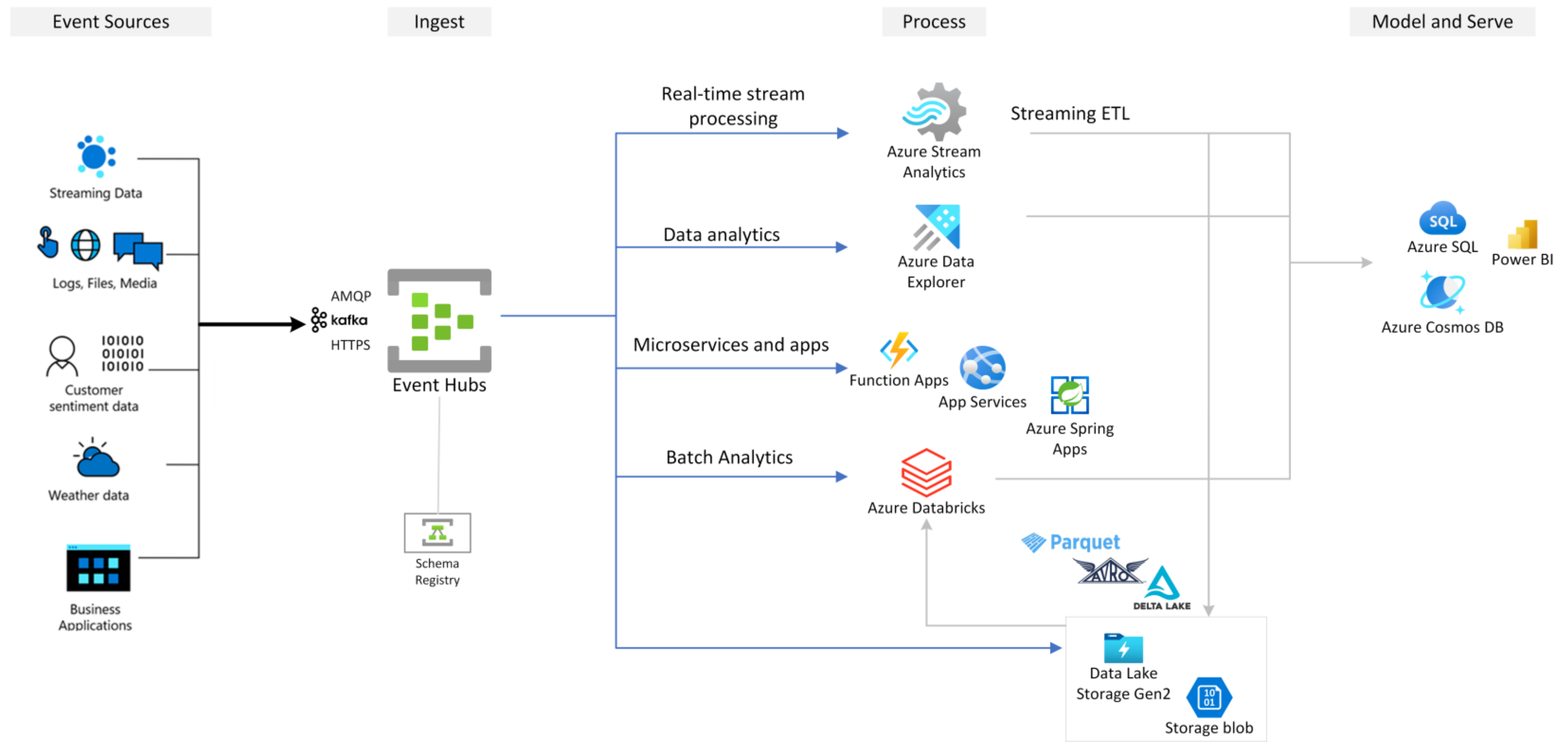
Scenario: Request-Response vs. Pub-Sub





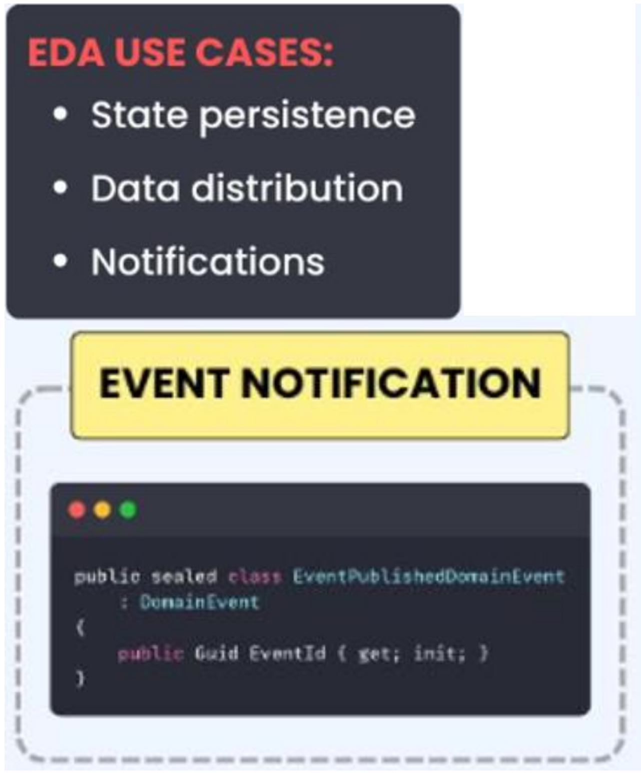
Event Streams

- Event Streams can be processed continuously, or in batch mode
  - An event can be processed when it is published (almost immediately)
  - A client can index through a stream of logged events (retrospectively)



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about

Event Driven Architectural Patterns

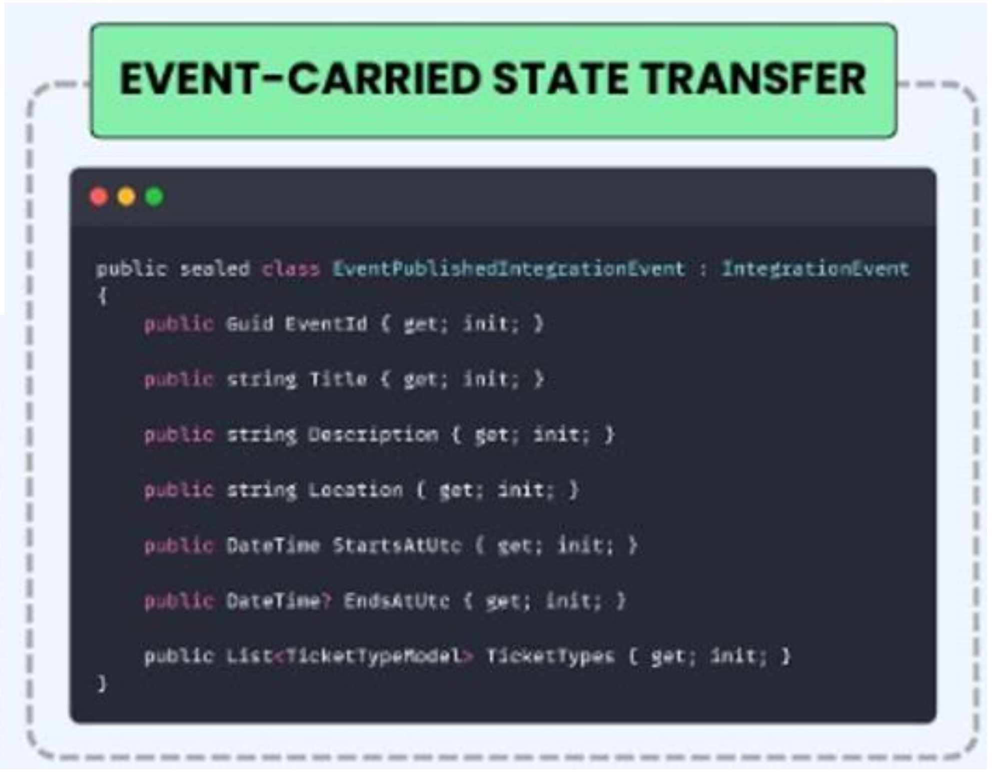


EDA USE CASES:

- State persistence
- Data distribution
- Notifications

EVENT NOTIFICATION

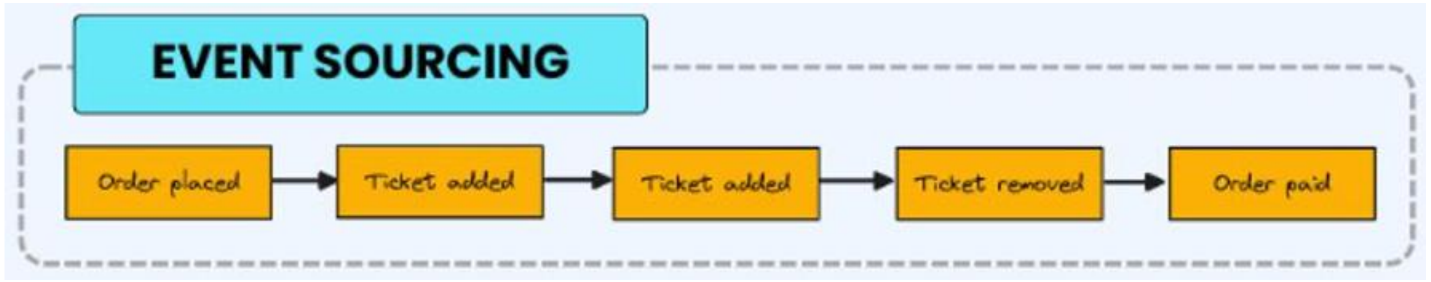
```
public sealed class EventPublishedDomainEvent
    : DomainEvent
{
    public Guid EventId { get; init; }
}
```

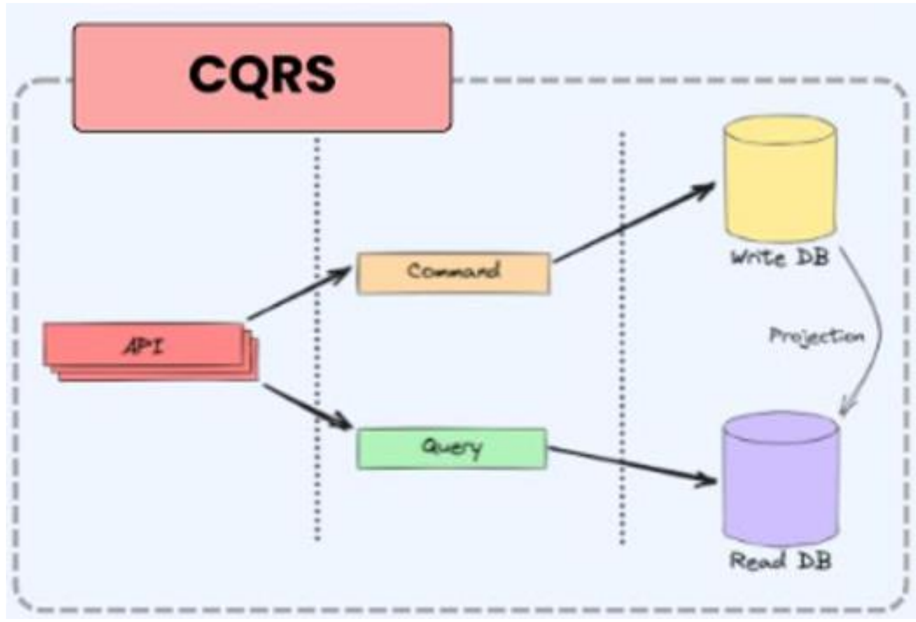


EVENT-CARRIED STATE TRANSFER

```
public sealed class EventPublishedIntegrationEvent : IntegrationEvent
{
    public Guid EventId { get; init; }

    public string Title { get; init; }

    public string Description { get; init; }

    public string Location { get; init; }

    public DateTime StartsAtUtc { get; init; }

    public DateTime? EndsAtUtc { get; init; }

    public List<TicketTypeModel> TicketTypes { get; init; }
}
```

Milan Jovanović, LinkedIn: “Patterns of Event-Driven Architecture”



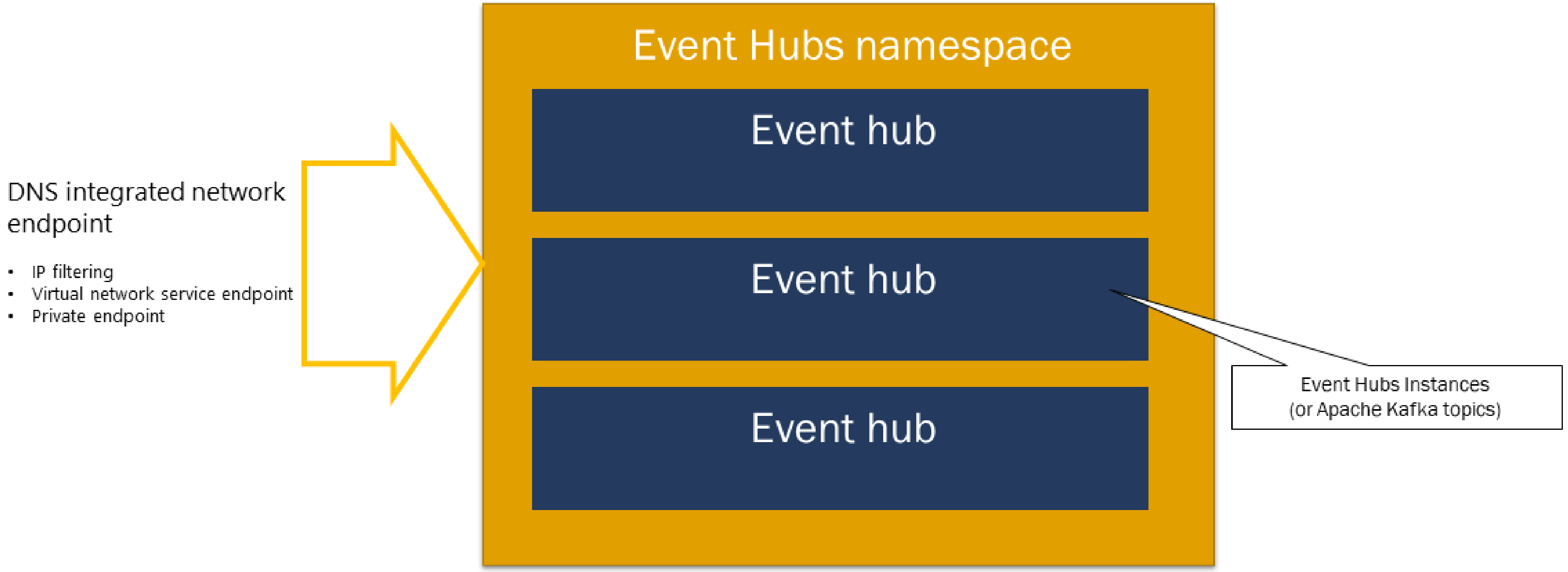


Azure Event Hub

Namespaces, Partitions, Consumer Groups, Checkpointing and Throughput Units

Event Hubs Namespace

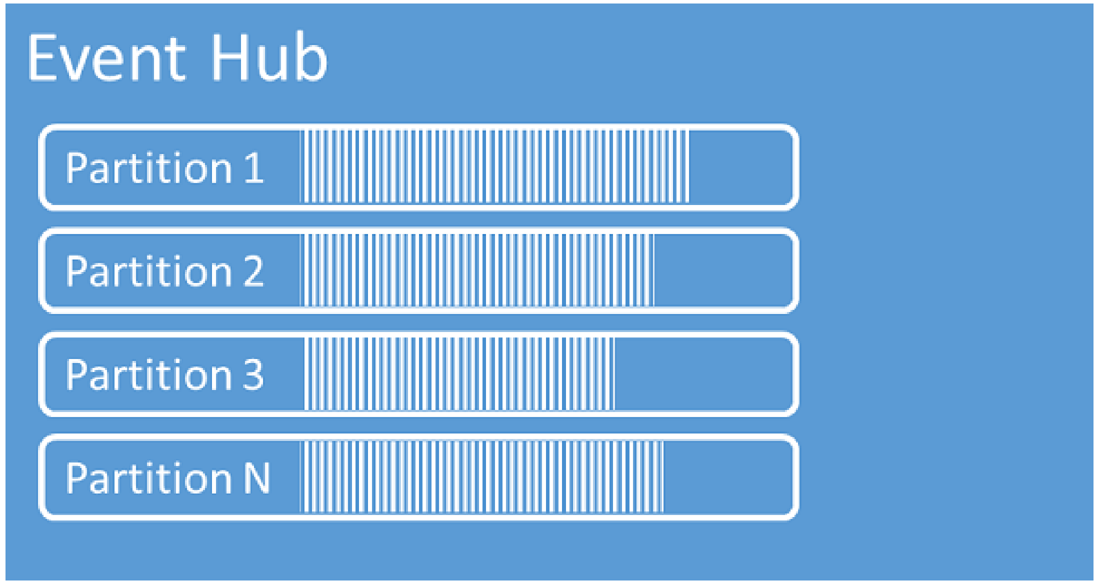
- Namespace: management container for Event Hubs
  integration management features


DNS integrated network endpoint
IP filtering
Virtual network service endpoint
Private endpoint
Event Hubs namespace
Event hub
Event hub
Event hub
Event Hubs Instances (or Apache Kafka topics)

https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Event Hubs Partitions

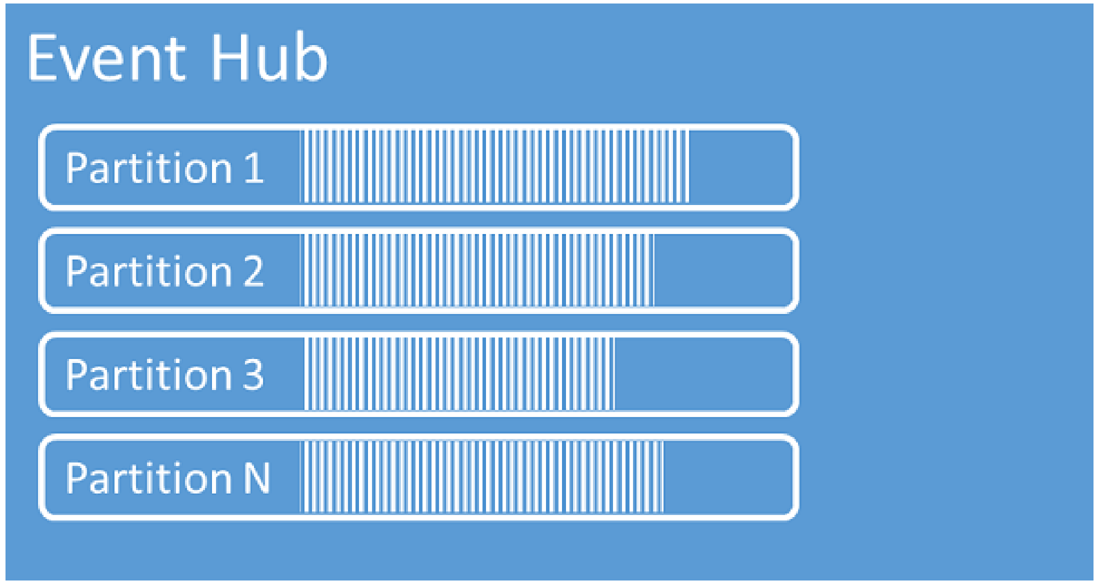
- Partitions aid throughput.
- Event Hub organises received events into one or more partitions. As newer events arrive, they're added to the end of this sequence.
- A partition can be thought of as a commit log. Partitions hold event data that contains the following information:
    - Body of the event
    - User-defined property bag describing the event
    - Metadata such as its offset in the partition, its number in the stream sequence
    - Service-side timestamp at which it was accepted



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Event Hubs Partitions: Considerations

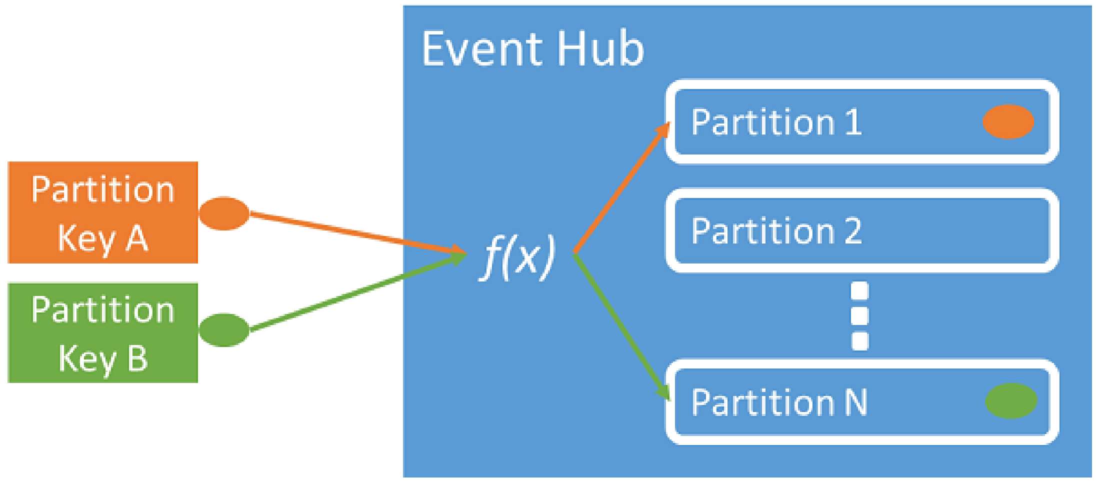
- The number of partitions is specified at the time of creating an event hub.
  - Pricing is not dependent on the number of partitions
  - Maximum number is tempting – but can make processing more complex
    - Choose enough to meet the peak load of your application for that particular event hub.
  - Number *can* be increased for premium, dedicated tiers
    - Changes event distribution!



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Publishing Events to Partitions

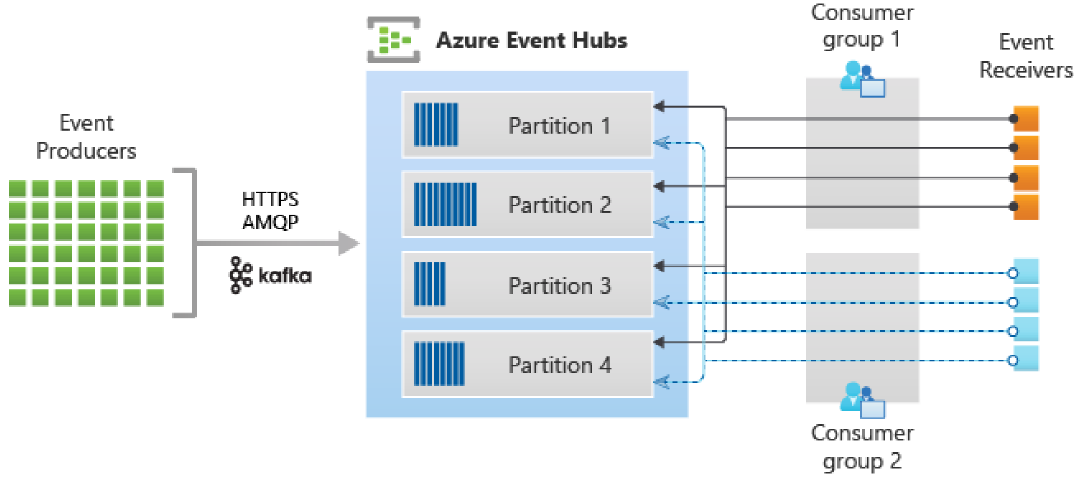
- *Can* publish direct to a specific partition, but not recommended.
- Partition Key is sender-supplied, hashed for a mapping.
  - Keeps related events together, in order
- In absence of a partition key, round-robin assignment applied



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Consuming events

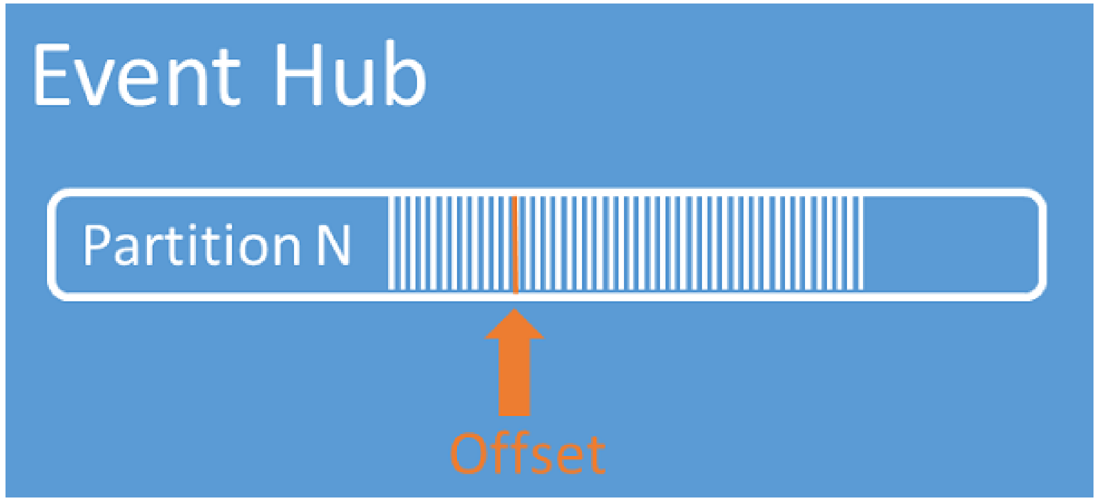
- Any entity that reads event data from an event hub is an *event consumer*.
  - All Event Hubs consumers connect via the AMQP 1.0 session and events are delivered through the session as they become available. (The client doesn't need to poll for data availability)
- A *consumer group* is a logical grouping of consumers
  - Consumer Groups enable multiple consuming applications to read the same streaming data in an event hub independently at their own pace with their offsets.



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Offsets and Checkpointing

- An *offset* is the position of an event within a partition
  - Effectively, a client-side cursor
  - Enables an event consumer to specify a point from which to begin reading events
- *Checkpointing* is a process by which readers mark or commit their position within a partition event sequence.
  - If a reader disconnects from a partition, when it reconnects it begins reading at the checkpoint that was previously submitted
  - Checkpointing is the responsibility of the consumer and occurs on a per-partition basis within a consumer group.



https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features

Throughput Units (TU)

- Throughput units are pre-purchased units of capacity. A single throughput unit:
    - Ingress: Up to 1 MB per second or 1,000 events per second (whichever comes first).
    - Egress: Up to 2 MB per second or 4,096 events per second.
- Beyond the capacity of the purchased throughput units:
    - Ingress is throttled and Event Hubs throws a ServerBusyException.
    - Egress doesn't produce throttling exceptions but is still limited to the capacity of the purchased throughput units.
- Throughput units are billed per hour for a minimum of one hour.
    - Up to 40 throughput units can be purchased for an Event Hubs namespace and are shared across all event hubs in that namespace
- The Auto-inflate feature of Event Hubs automatically scales up by increasing the number of throughput units, to meet usage needs.

https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-scalability#throughput-units

Event Hubs clients

Event Hub clients: why so many?

- Event Hub is complex
  - Clients for simple usage, through to more specialised cases

- Different areas of functionality for Event Hubs
  - Clients provided for a concrete set of scenarios

- Grouped into “Mainstream” and “Specialized”

- https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/eventhub/Azure.Messaging.EventHubs/samples/Sample02_EventHubsClients.md

Mainstream Event Hub clients

- EventHubBufferedProducerClient
  - Abstracts event batching and sending for simple usage.
- EventHubProducerClient
  - Caller has responsibility for batching and sending.

- EventHubConsumerClient
  - Simple read from one or all partitions (not recommended for Production)
- EventProcessorClient
  - Requires rigour suitable for Production to read from partitions

- https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/eventhub/Azure.Messaging.EventHubs/samples/Sample02_EventHubsClients.md

Specialized Event Hub clients

- PartitionReceiver
  - More control over reading from a specific partition
- PluggableCheckpointStoreEventProcessor<TPartition>
  - Base for custom reading/processing events, with checkpointing and greater control of communication with Event Hubs
- EventProcessor<TPartition>
  - Lowest-level, abstract base for custom processor, with most customisation available.

- https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/eventhub/Azure.Messaging.EventHubs/samples/Sample02_EventHubsClients.md

PoC: Event Streaming, Triggering an Azure Function

Demonstration of concepts using “Mainstream” clients

PoC purpose

- Demonstration 1 (Event Streams):
  - Publishing Events to Event Hub
  - Retrospectively reading Event Streams from Event Hub
  - Reading the Partition ID on receipt
  - Use of Consumer Groups
  - Checkpointing

- Demonstration 2 (Events as a Trigger):
  - Selecting a Partition when publishing Events to Event Hub
  - Triggering a Function on Events from an Event Hub

PoC components

Common components

1 Event Hub, with:

- 2 Partitions *(0, 1)*
- 2 Consumer Groups *($Default, consumergroup2*).

PostMan

- Used to interact with the Web APIs

Demonstration 1

3 Web APIs:

- 1 to publish Event Streams (*EventStreamPublisher*)
- 2 to read event streams (*EventStreamReaderNoCheckpoint*, *EventStreamReaderCheckpoint*)

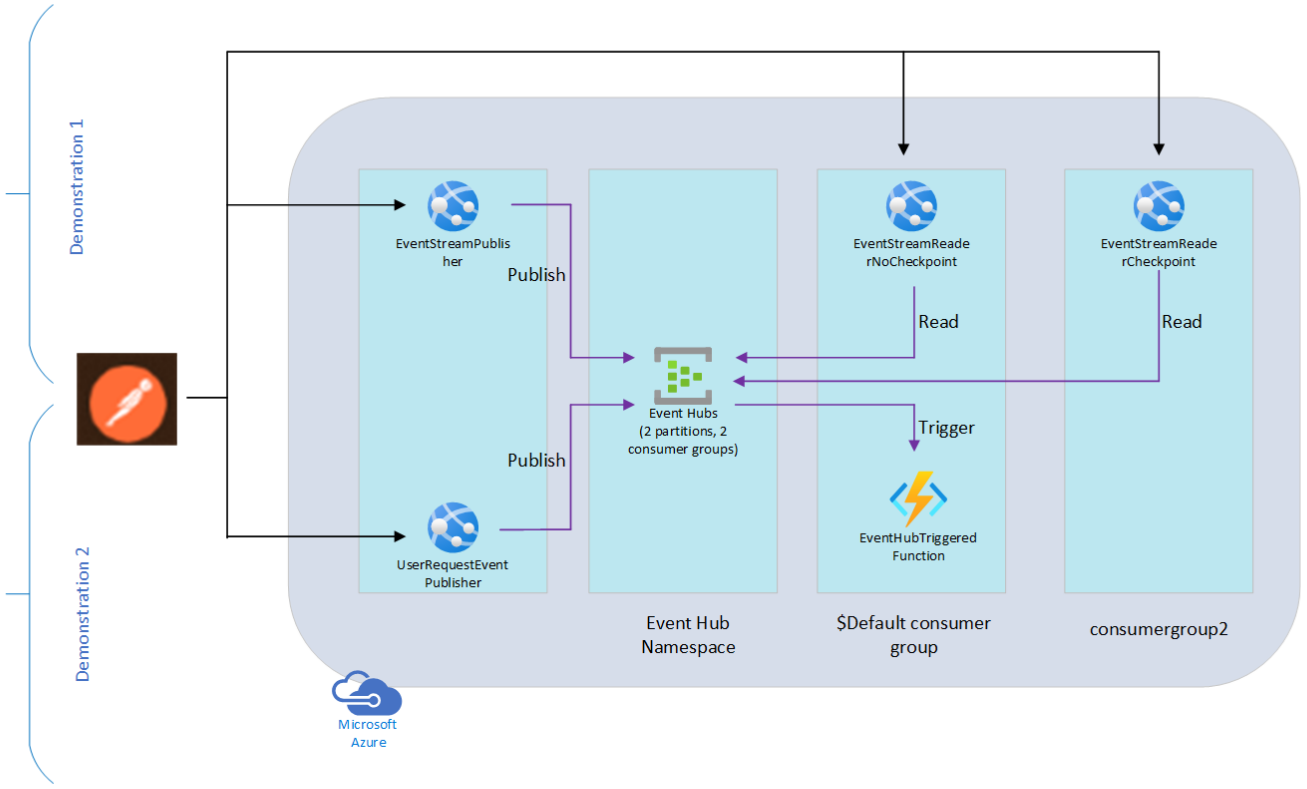
Demonstration 2

1 Web API:

- To publish a discrete Event on receipt of a user Request (*UserRequestEventPublisher*)

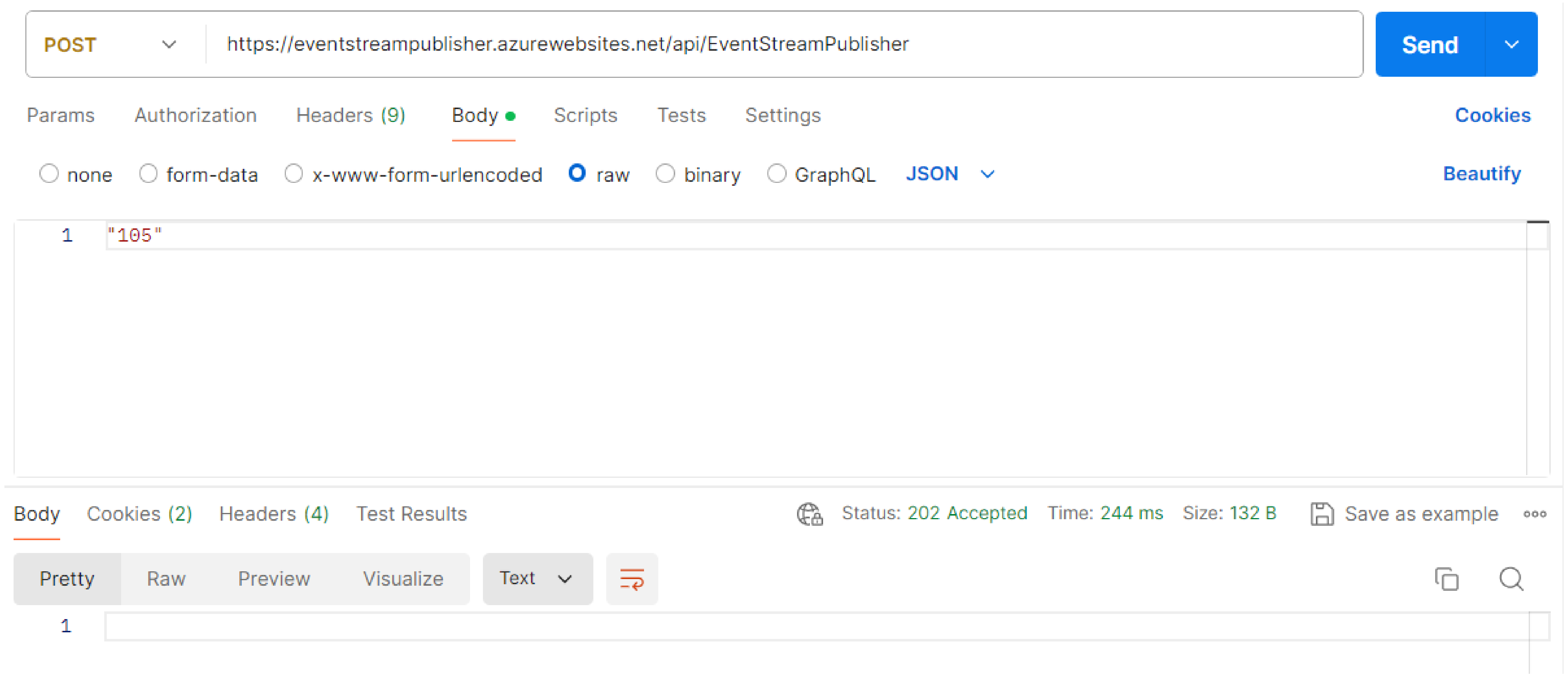
1 Function

- Triggered by events from Event Hub


Demonstration 1
Demonstration 2
EventStreamPublisher
Publish
Publish
UserRequestEvent Publisher
Event Hubs (2 partitions, 2 consumer groups)
EventStreamReaderNoCheckpoint
Read
Trigger
EventHubTriggered Function
EventStreamReaderCheckpoint
Read
Event Hub Namespace
$Default consumer group
consumergroup2
Microsoft Azure

Demonstration 1 (Event Stream) Views

Event Stream Publisher



POST https://eventstreampublisher.azurewebsites.net/api/EventStreamPublisher

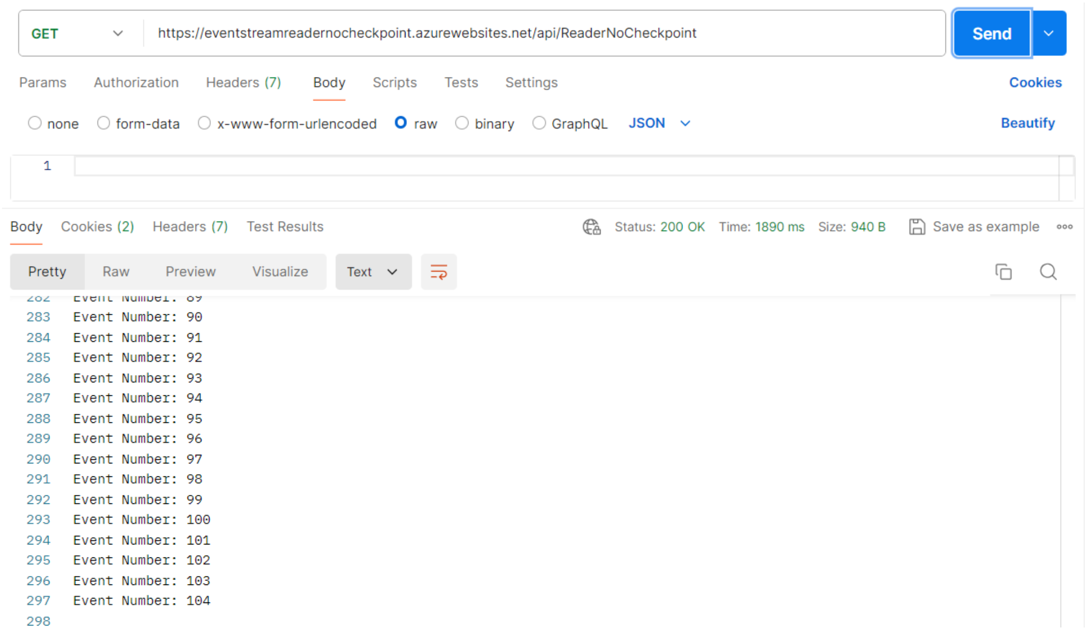
```
"105"
```

Status: 202 Accepted Time: 244 ms Size: 132 B

Body: Text

Event Stream Readers

No Checkpoint



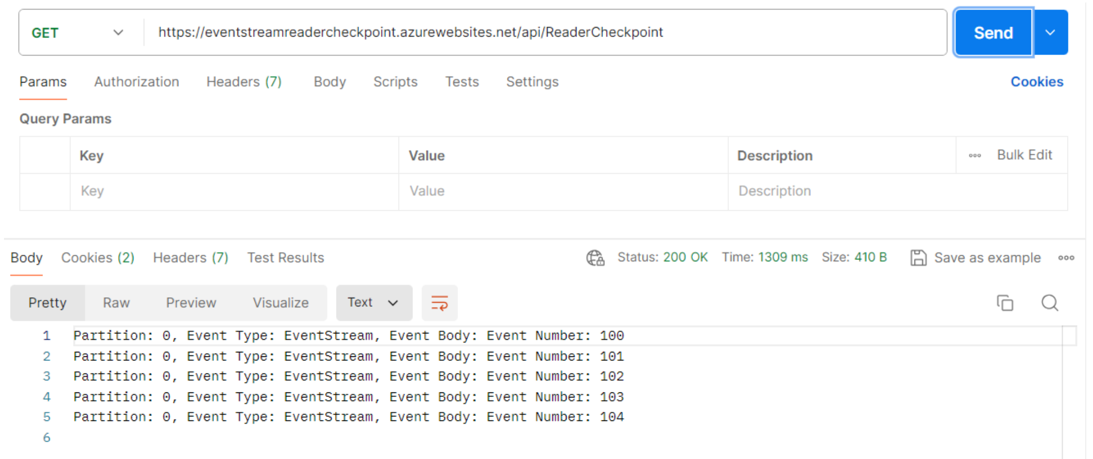
GET https://eventstreamreadernocheckpoint.azurewebsites.net/api/ReaderNoCheckpoint

Status: 200 OK Time: 1890 ms Size: 940 B

```
Event Number: 89
Event Number: 90
Event Number: 91
Event Number: 92
Event Number: 93
Event Number: 94
Event Number: 95
Event Number: 96
Event Number: 97
Event Number: 98
Event Number: 99
Event Number: 100
Event Number: 101
Event Number: 102
Event Number: 103
Event Number: 104
```

Without checkpointing, all events are retrieved for each subsequent call

Checkpoint



GET https://eventstreamreadercheckpoint.azurewebsites.net/api/ReaderCheckpoint

Status: 200 OK Time: 1309 ms Size: 410 B

```
Partition: 0, Event Type: EventStream, Event Body: Event Number: 100
Partition: 0, Event Type: EventStream, Event Body: Event Number: 101
Partition: 0, Event Type: EventStream, Event Body: Event Number: 102
Partition: 0, Event Type: EventStream, Event Body: Event Number: 103
Partition: 0, Event Type: EventStream, Event Body: Event Number: 104
```

With checkpointing, set at 10, in the second call only those events that exceed the checkpoint are retrieved. (Partition ID and Event Type also displayed)

Demonstration 1: Event Streams

Demonstration 1 (Event Stream) Construction

Storage Containers for Stream Readers

- Create a Storage Account
  - Standard
  - LRS
- Create a container for EventStreamReaderCheckpoint
- Create a container for EventStreamReaderNoCheckpoint

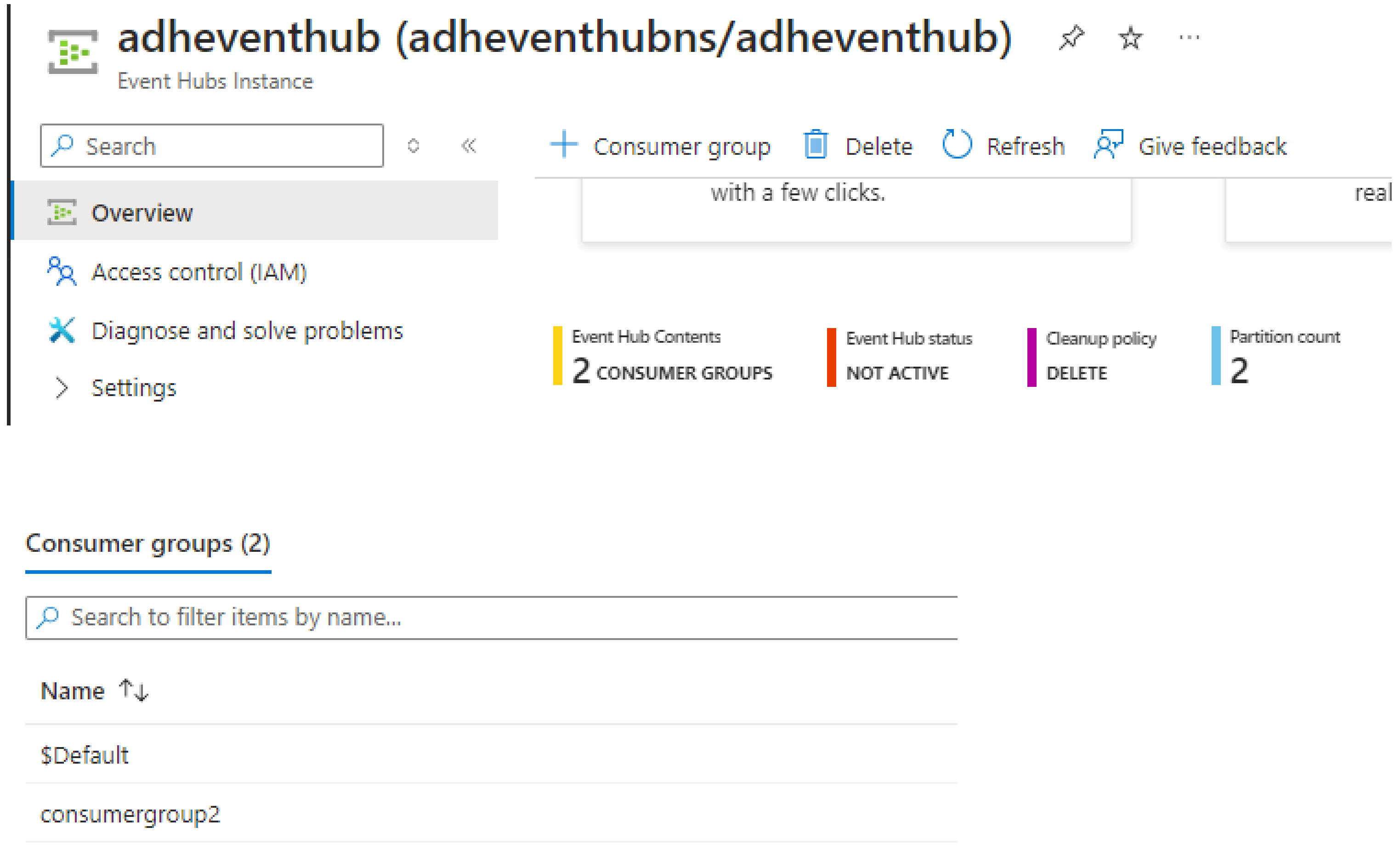
Create an Event Hub Namespace

- Create an “Event Hubs” (Namespace)
  - Standard Tier (Basic Tier supports one Consumer Group, only)
  - 1 Throughout Unit
- Create a new Shared access policy for the Event Hub Namespace:
  - “listener”, with Listen Claims
  - The “listen” access is required at Namespace scope

Create an Event Hub instance

- Create an Event Hub (instance) in the Namespace
    - 2 Partitions
- Once the Event Hub instance is created
    - Add a second Consumer Group (named consumergroup2)
    - Create a Shared access policy, “sender”, with Send claims
        - An Event Source can send to individual Event Hub instances

Event Hub, in Event Hubs Namespace



adheventhub (adheventhubns/adheventhub)

Event Hubs Instance

with a few clicks.

real

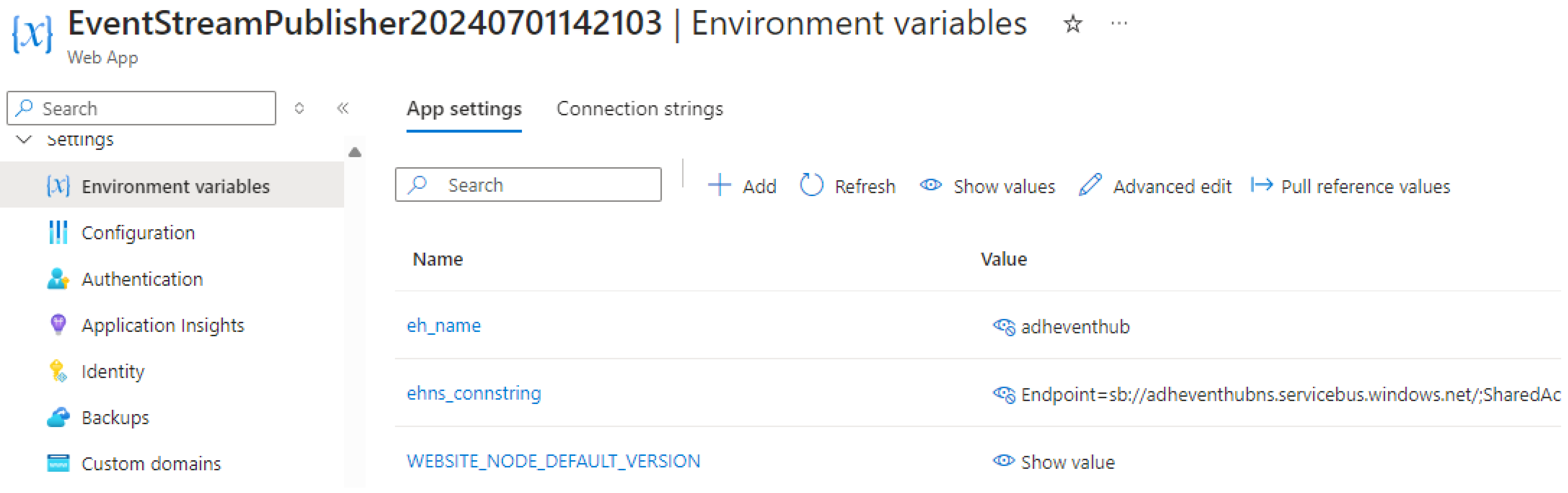
Event Hub Contents	Event Hub status	Cleanup policy	Partition count
2 CONSUMER GROUPS	NOT ACTIVE	DELETE	2

Consumer groups (2)

Search to filter items by name...

Name ↑↓
$Default
consumergroup2

EventStreamPublisher



EventStreamPublisher20240701142103 | Environment variables

Web App

Name	Value
eh_name	adheventhub
ehns_connstring	Endpoint=sb://adheventhubns.servicebus.windows.net/;SharedAc
WEBSITE_NODE_DEFAULT_VERSION	Show value

“ehns_connstring” set as the connection string for the “sender” policy of the Event Hub instance (;EntityPath=*Event Hub name)* is appended
“eh_name” set as the name of the Event Hub instance

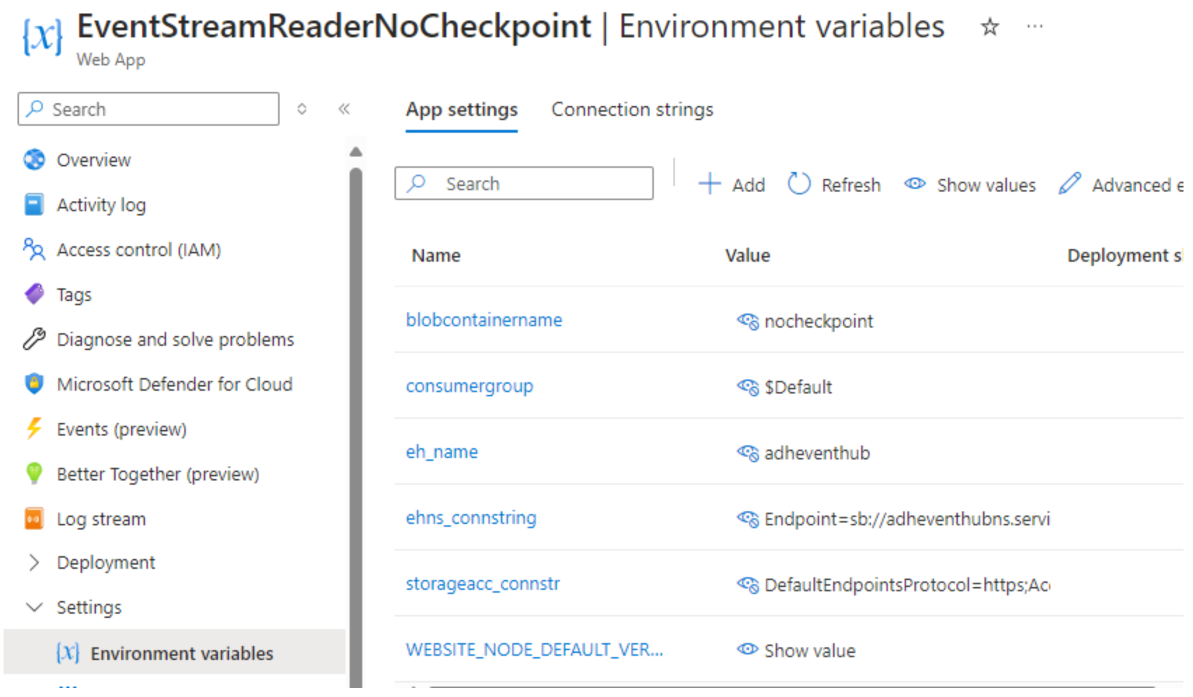
EventStreamPublisher code

- Controller has a POST endpoint
  - Takes a string expressing an integer value
  - Launches the service method “SendNEvents” with the integer value, in a “fire and forget” manner
  - Returns status code 202 (Accepted)
- Service
  - Creates instance of the SDK class EventHubProducerClient (using “eh_name”, “ehns_connstring”)
  - Method “SendNEvent” creates a batch of the input number of events, then awaits the EventHubProducerClient method “SendAsync” acting on the batch

- https://github.com/ahironatava/AzureEventHubExample/tree/main/EventStreamPublisher/EventStreamPublisher

EventStreamReaderNoCheckpoint

- “consumergroup” – the Event Hub Consumer Group to use ($Default)
- “blobcontainername” – the name of the associated container
- “storageacc_connstr” – the Storage Account Connection String
- “eh_name” - the name of the Event Hub instance
- “ehns_connstring” – connection string for the “listener” policy of the Event Hub Namespace, with ;EntityPath=*Event Hub name* appended



Name	Value	Deployment s
blobcontainername	nocheckpoint	
consumergroup	$Default	
eh_name	adheventhub	
ehns_connstring	Endpoint=sb://adheventhubns.servi	
storageacc_connstr	DefaultEndpointsProtocol=https;Ac	
WEBSITE_NODE_DEFAULT_VER...	Show value	

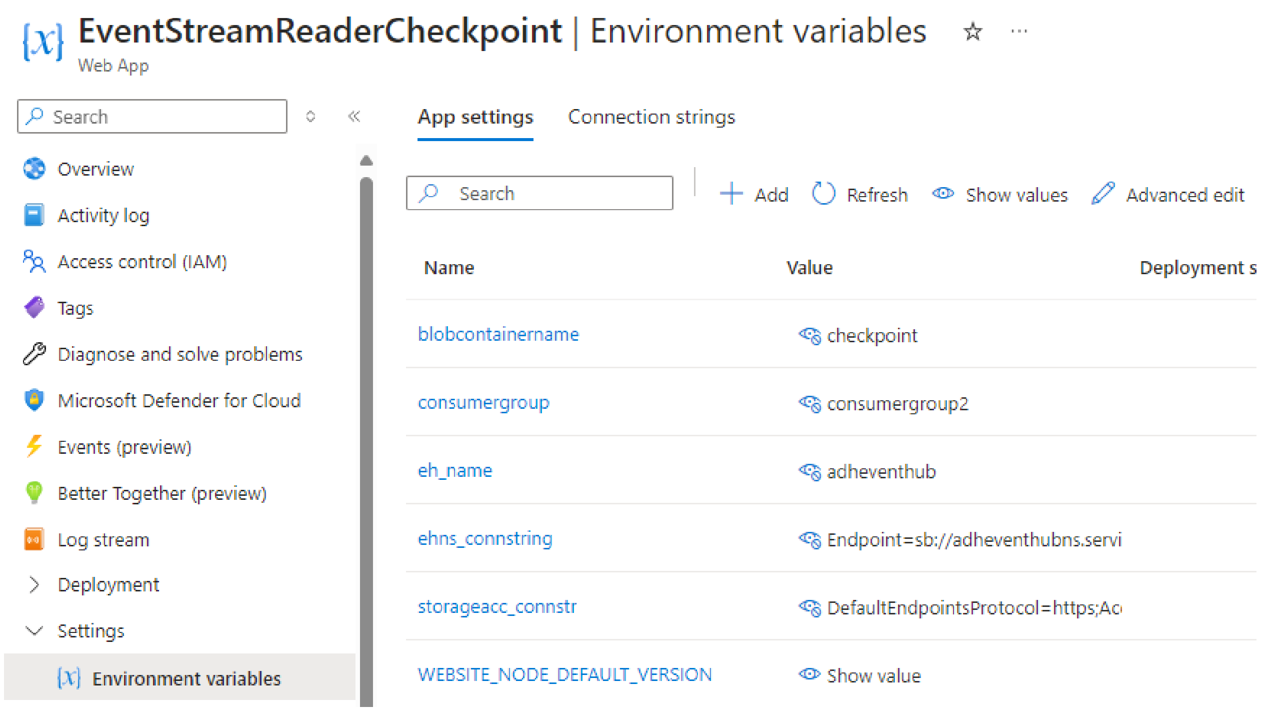
EventStreamReaderNoCheckpoint code

- Controller has 2 GET endpoints
  - First endpoint takes an integer, the second utilises a default value (1)
  - Both call the service method “GetEntriesInTimebox” with the integer value
  - Returns status code 200 (OK) and the string of entries returned by the service
- Service
  - Constructor
    - Creates a BlobContainerClient instance (using “blobcontainername” and “storageacc_connstr”)
    - Creates EventProcessorClient instance (using the Blob Container Client, “eh_name”, “ehns_connstring”, and “consumergroup”)
    - Assigns handlers “ProcessEventHandler” and “ProcessErrorHandler”
  - Method “GetEntriesInTimebox” enables the Event Processor Client for the specified timebox and returns an accumulated list of strings
    - ProcessEventHandler : All successfully received events are appended as strings to the accumulated list
    - ProcessErrorHandler: Errors are appended as strings to the accumulated list

- https://github.com/ahironatava/AzureEventHubExample/tree/main/EventStreamReaderNoCheckpoint

EventStreamReaderCheckpoint

- Similar to settings for EventStreamReaderNoCheckpoint
- Different setting for "consumergroup" (consumergroup2)


EventStreamReaderCheckpoint | Environment variables
Web App

Name	Value	Deployment s
blobcontainername	checkpoint	
consumergroup	consumergroup2	
eh_name	adheventhub	
ehns_connstring	Endpoint=sb://adheventhubns.servi	
storageacc_connstr	DefaultEndpointsProtocol=https;Ac	
WEBSITE_NODE_DEFAULT_VERSION	Show value	

EventStreamReaderCheckpoint code

- Controller is very similar to EventStreamReaderNoCheckpoint, with 2 GET endpoints that call the “GetEntriesInTimebox” method
- Service is also similar but in addition applies checkpointing
  - Constructor initialises Checkpointing
    - Determines the threshold number of events to process before applying a checkpoint
    - Creates a dictionary (key = Partition ID, value = event count)
  - ProcessEventHandler
    - Appends the string as an event to the accumulated list
    - Updates checkpointing: increments the count of events for the partition and if this exceeds the threshold it applies a checkpoint, then resets the count to zero.

- https://github.com/ahironatava/AzureEventHubExample/tree/main/EventStreamReaderCheckpoint/EventStreamReaderCheckpoint

Demonstration 2 (Events as Trigger) Views

PoC components

Common components

1 Event Hub, with:

- 2 Partitions (*0, 1*)
- 2 Consumer Groups (*$Default, consumergroup2*).

PostMan

- Used to interact with the Web APIs

Demonstration 1

3 Web APIs:

- 1 to publish Event Streams (*EventStreamPublisher*)
- 2 to read event streams (*EventStreamReaderNoCheckpoint*, *EventStreamReaderCheckpoint*)

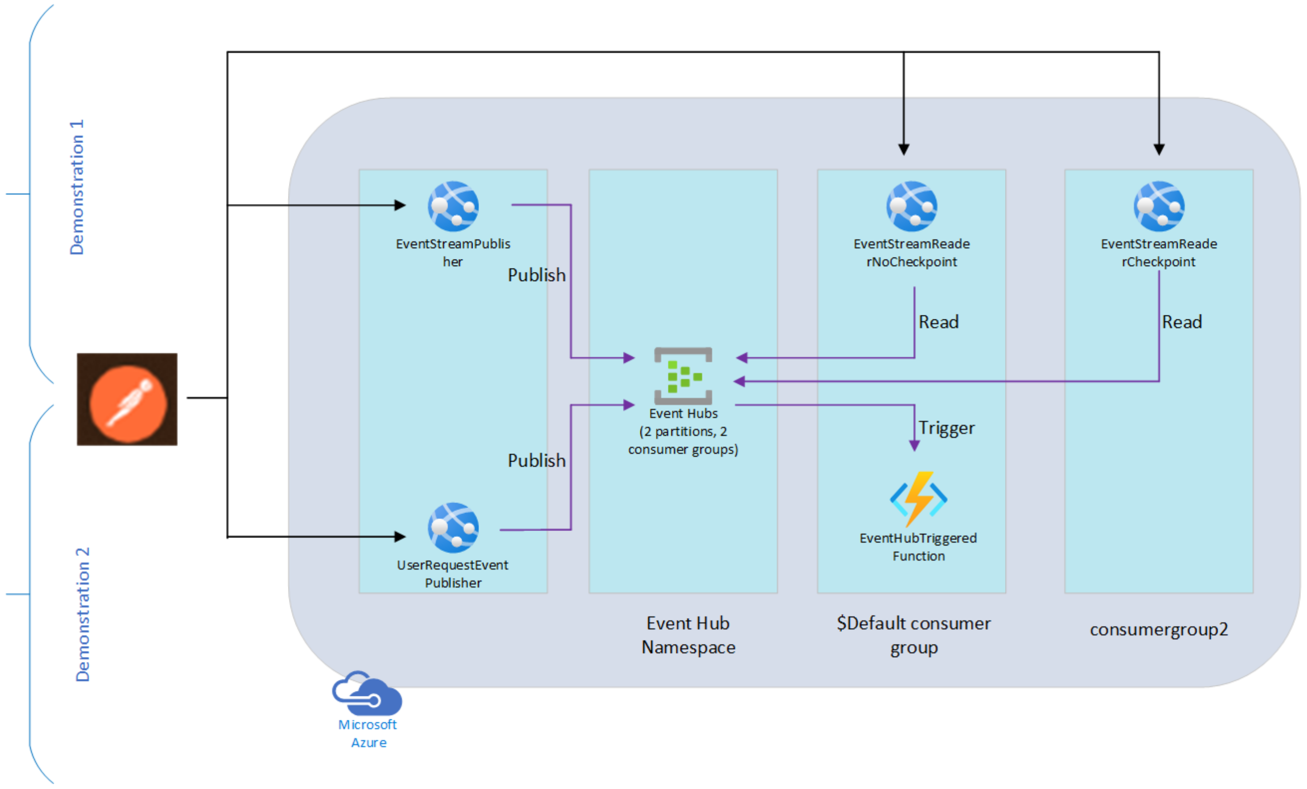
Demonstration 2

1 Web API:

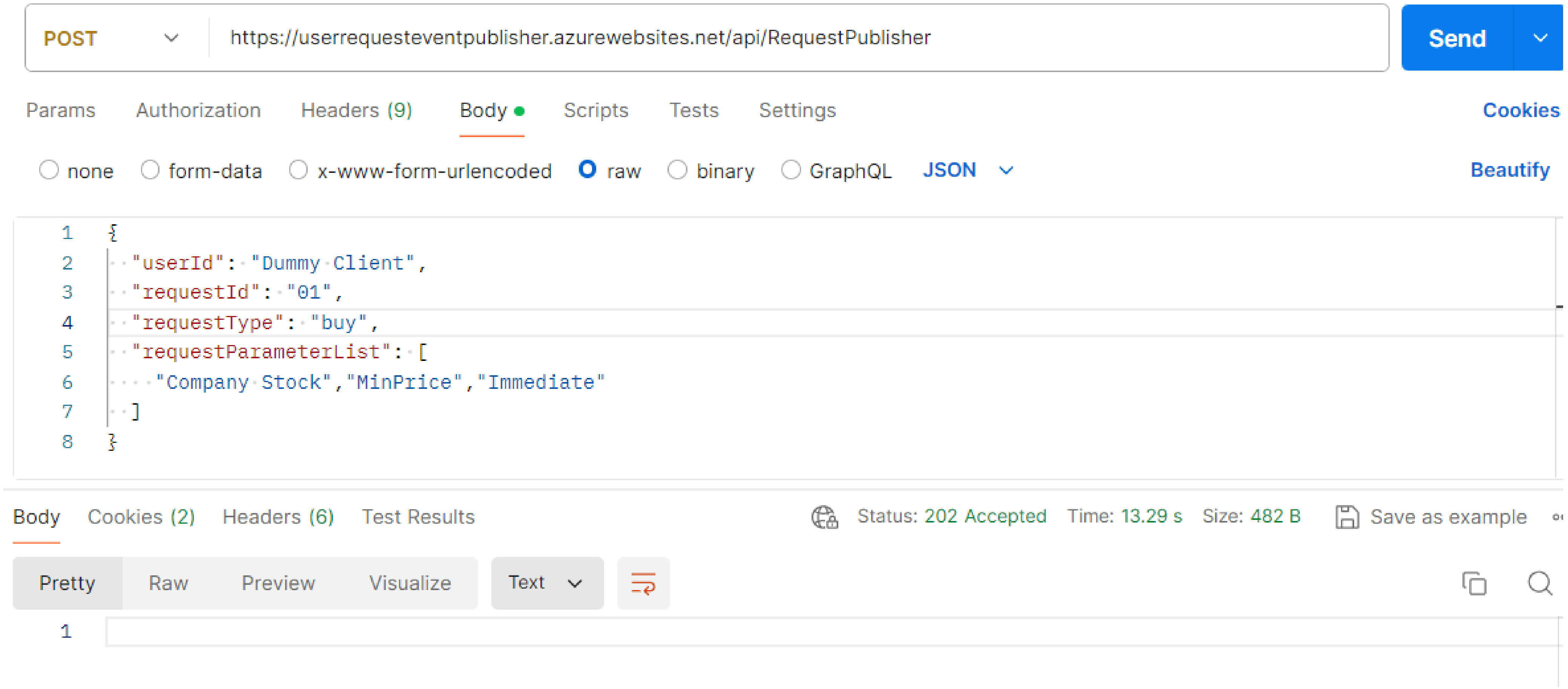
- To publish a discrete Event on receipt of a user Request (*UserRequestEventPublisher*)

1 Function

- Triggered by events from Event Hub


Demonstration 1
Demonstration 2
EventStreamPublisher
Publish
UserRequestEvent Publisher
Publish
Event Hubs (2 partitions, 2 consumer groups)
EventStreamReaderNoCheckpoint
Read
Trigger
EventHubTriggered Function
EventStreamReaderCheckpoint
Read
Event Hub Namespace
$Default consumer group
consumergroup2
Microsoft Azure

Request Publisher



POST https://userrequesteventpublisher.azurewebsites.net/api/RequestPublisher

Body: ( ) none ( ) form-data ( ) x-www-form-urlencoded [✓ raw] ( ) binary ( ) GraphQL JSON

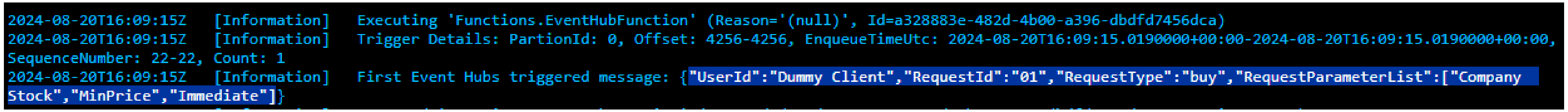
```
{
  "userId": "Dummy Client",
  "requestId": "01",
  "requestType": "buy",
  "requestParameterList": [
    "Company Stock","MinPrice","Immediate"
  ]
}
```

Status: 202 Accepted Time: 13.29 s Size: 482 B

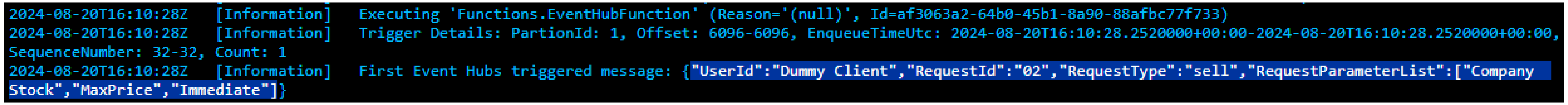
```
1
```

Event Hub Triggered Function, Log Stream

UserRequestPublisher Controller (POST) receives a UserRequest:
- Sends “sell” requests to Partition 1 of the Event Hub
- Otherwise, default partition (0).



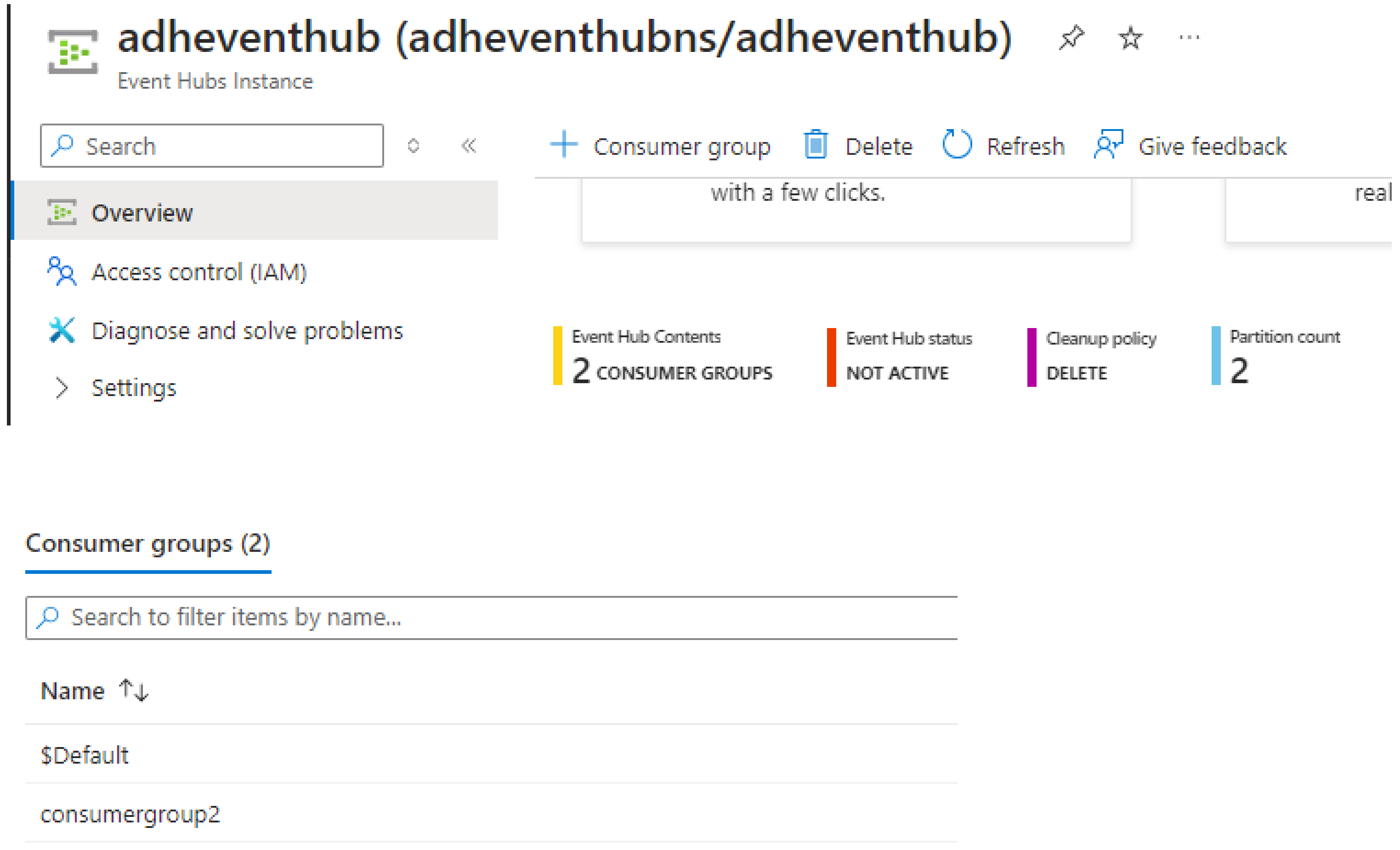
```
2024-08-20T16:09:15Z   [Information]   Executing 'Functions.EventHubFunction' (Reason='(null)', Id=a328883e-482d-4b00-a396-dbdfd7456dca)
2024-08-20T16:09:15Z   [Information]   Trigger Details: PartionId: 0, Offset: 4256-4256, EnqueueTimeUtc: 2024-08-20T16:09:15.0190000+00:00-2024-08-20T16:09:15.0190000+00:00,
SequenceNumber: 22-22, Count: 1
2024-08-20T16:09:15Z   [Information]   First Event Hubs triggered message: {"UserId":"Dummy Client","RequestId":"01","RequestType":"buy","RequestParameterList":["Company
Stock","MinPrice","Immediate"]}
```



```
2024-08-20T16:10:28Z   [Information]   Executing 'Functions.EventHubFunction' (Reason='(null)', Id=af3063a2-64b0-45b1-8a90-88afbc77f733)
2024-08-20T16:10:28Z   [Information]   Trigger Details: PartionId: 1, Offset: 6096-6096, EnqueueTimeUtc: 2024-08-20T16:10:28.2520000+00:00-2024-08-20T16:10:28.2520000+00:00,
SequenceNumber: 32-32, Count: 1
2024-08-20T16:10:28Z   [Information]   First Event Hubs triggered message: {"UserId":"Dummy Client","RequestId":"02","RequestType":"sell","RequestParameterList":["Company
Stock","MaxPrice","Immediate"]}
```

Demonstration 2 (Events as Trigger) Construction

Event Hub, in Event Hubs Namespace



adheventhub (adheventhubns/adheventhub)

Event Hubs Instance

with a few clicks.

real

Event Hub Contents	Event Hub status	Cleanup policy	Partition count
2 CONSUMER GROUPS	NOT ACTIVE	DELETE	2

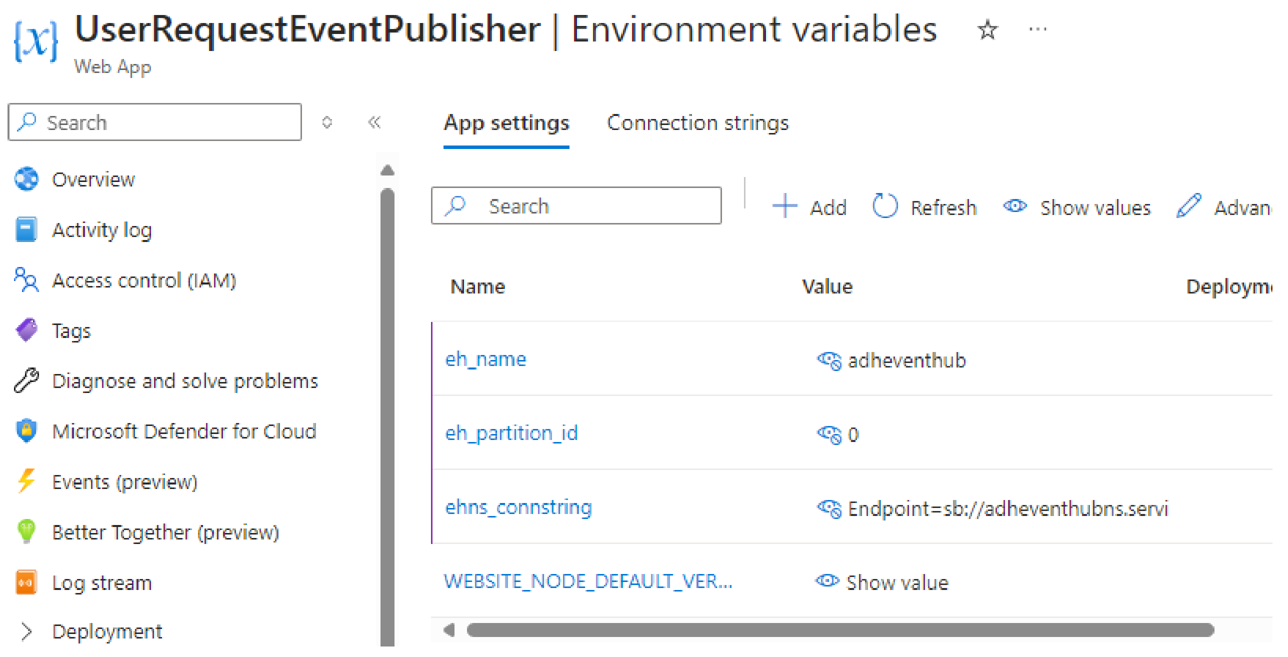
Consumer groups (2)

Search to filter items by name...

Name ↑↓
$Default
consumergroup2

UserRequestEventPublisher

- “eh_name” - the name of the Event Hub instance
- “eh_partition_id” – the *default* partition to use (0)
- “ehns_connstring” – connection string for the “sender” policy of the Event Hub (includes *;EntityPath=Event Hub name* at the end)


UserRequestEventPublisher | Environment variables
Web App

Name	Value	Deploym
eh_name	adheventhub	
eh_partition_id	0	
ehns_connstring	Endpoint=sb://adheventhubns.servi	
WEBSITE_NODE_DEFAULT_VER...	Show value	

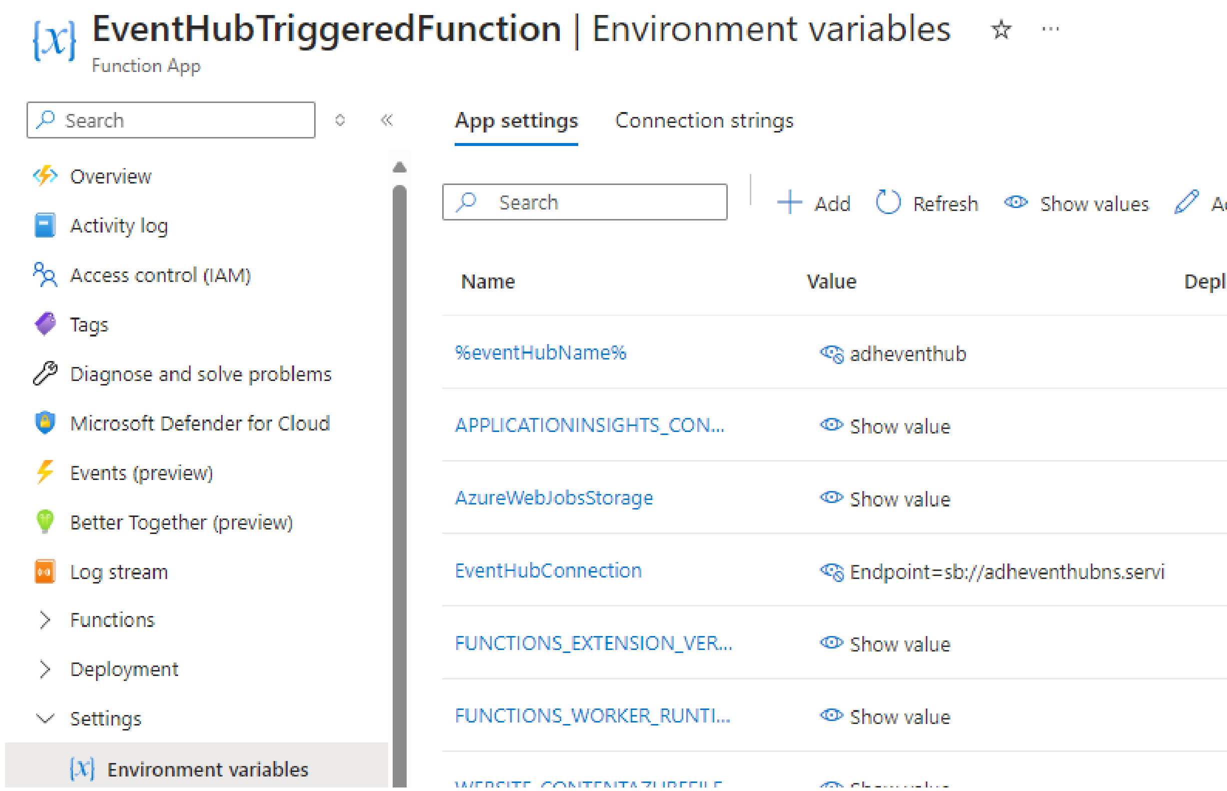
UserRequestEventPublisher code

- Controller has a POST endpoint
  - Takes a “UserRequest” object
  - Launches the service method “ProcessRequest” with the provided object, in a “fire and forget” manner
  - Returns status code 202 (Accepted)
- Service
  - Constructor
    - Creates Event Hub Producer Client (using “eh_name”, “ehns_connstring”)
    - Sets default partition ID to 0, if not specified in configuration (“eh_partition_id”)
  - Method “ProcessRequest”
    - Updates the partition ID to 1 if the object’s value for “RequestType” is “sell“ (set to default of 0, otherwise)
    - Creates EventData from the provided object, setting “EventType” to “UserRequest” (for consumer filters)
    - Awaits the Event Hub Producer Client method “SendAsync” acting on a List containing the event data

- https://github.com/ahironatava/AzureEventHubExample/tree/main/UserRequestEventPublisher

EventHubTriggeredFunction

- ~~“%eventHubName%” - the name of the Event Hub instance~~
- “EventHubConnection” – connection string for the Namespace “listener” policy – with the following string appended (and substituting the event hub name): ;EntityPath=*EventHubName*



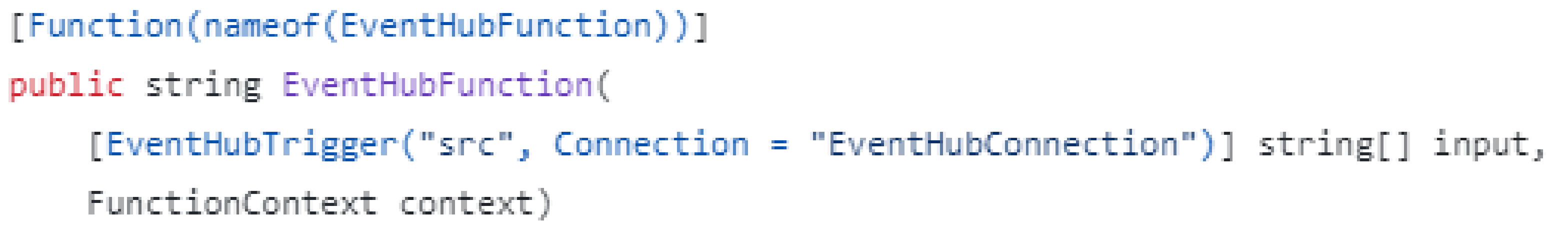
EventHubTriggeredFunction | Environment variables

Function App

Name	Value
%eventHubName%	adheventhub
APPLICATIONINSIGHTS_CON...	Show value
AzureWebJobsStorage	Show value
EventHubConnection	Endpoint=sb://adheventhubns.servi
FUNCTIONS_EXTENSION_VER...	Show value
FUNCTIONS_WORKER_RUNTI...	Show value

EventHubTriggeredFunction code

- Function1.cs is triggered by Event Hub events:



```
[Function(nameof(EventHubFunction))]
public string EventHubFunction(
    [EventHubTrigger("src", Connection = "EventHubConnection")] string[] input,
    FunctionContext context)
```

- Once triggered, the code logs the event as a string

- https://github.com/ahironatava/AzureEventHubExample/tree/main/EventHubTriggeredFunction

Azure Event Hub: some lessons learned

Disable / Enable Event Hub

- Event Hub consumes credits on your subscription
  - Disabling the Hub when not in use *reduces* the rate of credit consumption
- Beware the page refresh illusion in the Azure Portal
  - The status displayed for the Event Hub may be stale
- Remember to re-enable the Event Hub before use!

Sending to Azure Event Hub

- Sender-supplied Partition Keys are automatically hashed to map to a valid Partition
- If specifying an explicit Partition ID, the ID must be valid
  - E.g., for two Partitions, valid IDs are 0, 1
  - Specifying a value greater than will 1 will fail, silently
- In Production systems, Access Keys and Connection strings should be scoped to the individual Hub and be for Send, only
  - Unless greater privileges are truly required.

Consuming Events from Azure Event Hub

- If one Consumer in a Consumer Group applies a Checkpoint, then all Consumers in that Consumer Group will be affected.

- For an Azure Function triggered by the Event Hub, the connection string must be specified correctly; either:
  - “%eventHubName%” specifies the Event Hub name; or
  - The Event Hub name appears at the end of the connection string, and “%eventHubName%” is not specified
  - https://learn.microsoft.com/en-us/azure/azure-functions/functions-bindings-event-hubs-trigger?tabs=python-v2%2Cisolated-process%2Cnodejs-v4%2Cfunctionsv2%2Cextensionv5&pivots=programming-language-csharp#attributes

- During development, taking different elements offline (e.g. the above Function) can lead to a loss of synch and error messages relating to the Event Hub epoch
  - For development, it may be simplest to delete the Blob storage allocated to the Event Hub
    - The Blob will automatically be recreated and re-initialised