Background Information

What is Azure Event Hubs?

As described in Microsoft Docs, 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. To read more, see the full article here.

What are the key components of an event hub?

- Event Hubs namespace: The namespace contains the set of Event Hubs instances that are currently being managed.
- Event Hub: A set of partitions (below) that hold event data as it is processed by consumers.
- Partitions: As an Event Hub receives events it organizes them into partitions, this allows for replication and increases throughput capacity.
- Producers: Any entity that is responsible for publishing data to a specific Event Hub
- Consumers: Any entity that reads event data from an event hub. The Event Hubs service delivers events through a session as they become available

To read more about the components of an event hub see the documentation here and here.

What are Event Hubs used for?

Some examples of what event hubs can be used for are the following:

- Anomaly detection (fraud/outliers), for example, looking at credit card transactions and determining when fraudulent purchases are made, if a group of purchases on one card are all in different regions it is potentially abnormal behavior.
- Logging messages generated by application code, the messages can be generated by the web framework or from application code directly.
- Transaction processing - high volume processing of batches of transactions, especially independent transactions that can be processed in parallel. (Such as financial transactions or experimental data gathered by scientific instruments),

this processing needs to maintain integrity in the system and always keeping everything in a consistent state

- Device telemetry streaming - data is pushed automatically without worrying about polling, this is more efficient, continuous, scalable and allows better access to real-time data, it is also good for automation, optimization, preventing troubleshooting, and load balancing

What is a producer?

A producer is a client responsible for publishing event data to a specific Event Hub. Currently this is done by grouping sets of event data together in batches. Depending on the options specified when sending, event data may be automatically routed to an available partition or sent to a specifically requested partition.

Introducing the Streaming Producer

Publishing events using the **producer client**, as described briefly above, is optimized for high and consistent throughput scenarios, allowing applications to collect a set of events as a batch and publish in a single operation. Calling send on a batch of events is a deterministic method. When it returns, the events were either successfully sent or a failure was reported. When using the producer client, developers are expected to build and manage batches according to the needs of their application, allowing them to prioritize trade-offs between ensuring batch density, enforcing strict ordering of events, and publishing on a consistent and predictable schedule.

On the other hand, the primary goal of the **streaming producer** is to provide developers with the ability to publish events without the need to explicitly manage batch construction, population, or service operations. Events are collected as they are queued, organized into batches, and published by the streaming producer as batches become full or a certain amount of time has elapsed. When queuing events, developers may request automatic routing to a partition or explicitly control the partition in the same manner supported by the producer client, with the streaming producer managing the details of grouping events into the appropriate batches for publication.

What are the advantages of the streaming producer?

When applications need to process low frequency or sparse event streams, both the Event Hub producer client and legacy clients do not provide efficient functionality for publishing a single event. In order to avoid inefficiencies that can negatively impact throughput, developers need to include non-trivial overhead to their code to manage decisions around caching batches-in-progress, routing events to the correct batch, and publishing partial batches when events aren't being published frequently. With the streaming producer, applications can queue events into the producer as needed and the producer will take care of efficiently managing batches and publishing.