# Choose whether to use messages or events

The first thing to understand about a communication is whether it sends messages or events. This knowledge will help you choose the appropriate Azure service to use.

For each communication, consider the following question**: Does the sending component expect the communication to be processed in a particular way by the destination component?**

If the answer is yes, choose to use a **message**. If the answer is no, you may be able to use **events**.

## Message

In distributed applications, messages have the following characteristics:

* A message contains raw data, produced by one component, that will be consumed by another component.
* A message contains the data itself, not just a reference to that data.
* The sending component expects the message content to be processed in a certain way by the destination component. The integrity of the overall system may depend on both sender and receiver doing a specific job.

## Event

Events are lighter weight than messages, and are most often used for broadcast communications. The components sending the event are known as **publishers**, and receivers are known as **subscribers**.

Events have the following characteristics:

* An event is a lightweight notification that indicates that something happened.
* The event may be sent to multiple receivers, or to none at all.
* Events are often intended to "fan out," or have a large number of subscribers for each publisher.
* The publisher of the event has no expectation about the action a receiving component takes.
* Some events are discrete units and unrelated to other events.
* Some events are part of a related and ordered series.

# Message-based delivery with queues

## Azure Queue Storage

Queue storage is a service that uses Azure Storage to store large numbers of messages that can be securely accessed from anywhere in the world using a simple REST-based interface

Choose Queue storage if:

* You need an audit trail of all messages that pass through the queue.
* You expect the queue to exceed 80 GB in size.
* You want to track progress for processing a message inside of the queue.

***Settings for queues***

When you create a storage account that will contain queues, you should consider the following settings:

* Queues are only available as part of Azure general-purpose storage accounts (v1 or v2). You cannot add them to Blob storage accounts.
* The **Access tier** setting which is shown for StorageV2 accounts applies only to Blob storage and does not affect queues.
* You should choose a location that is close to either the source components or destination components or (preferably) both.
* Data is always replicated to multiple servers to guard against disk failures and other hardware problems. You have a choice of replication strategies: Locally Redundant Storage (LRS) is low-cost but vulnerable to disasters that affect an entire data center while Geo-Redundant Storage (GRS) replicates data to other Azure data centers. Choose the replication strategy that meets your redundancy needs.
* The performance tier determines how your messages are stored: Standard uses magnetic drives while Premium uses solid-state drives. Choose Standard if you expect peaks in demand to be short. Consider Premium if queue length sometimes becomes long and you need to minimize the time to access messages.
* Require secure transfer if sensitive information may pass through the queue. This setting ensures that all connections to the queue are encrypted using Secure Sockets Layer (SSL).

***Queue identity***

Every queue has a name that you assign during creation. The name must be unique within your storage account but doesn't need to be globally unique (unlike the storage account name).

The combination of your storage account name and your queue name uniquely identifies a queue.

***Access authorization***

Every request to a queue must be authorized and there are several options to choose from.

|  |  |
| --- | --- |
| **Authorization Type** | **Description** |
| Azure Active Directory | You can use role-based authentication and identify specific clients based on AAD credentials. |
| Shared Key | Sometimes referred to as an account key, this is an encrypted key signature associated with the storage account. Every storage account has two of these keys that can be passed with each request to authenticate access. Using this approach is like using a root password - it provides full access to the storage account. |
| Shared access signature | A shared access signature (SAS) is a generated URI that grants limited access to objects in your storage account to clients. You can restrict access to specific resources, permissions, and scope to a data range to automatically turn off access after a period of time. |

## Azure Service Bus Queues

Service Bus is a message broker system intended for enterprise applications. These apps often utilize multiple communication protocols, have different data contracts, higher security requirements, and can include both cloud and on-premises services.

Choose Service Bus queues if:

* You need an At-Most-Once delivery guarantee.
* You need a FIFO guarantee.
* You need to group messages into transactions.
* You want to receive messages without polling the queue.
* You need to provide a role-based access model to the queues.
* You need to handle messages larger than 64 KB but less than 256 KB.
* Your queue size will not grow larger than 80 GB.
* You would like to be able to publish and consume batches of messages.

## Azure Service Bus Topics

Azure Service Bus topics are like queues, but can have multiple subscribers. When a message is sent to a topic instead of a queue multiple components can be triggered to do their work.

Choose Service Bus Topics if:

* you need multiple receivers to handle each message

## What is a relay?

A relay is an object that performs synchronous, two-way communication between applications. Unlike queues and topics, it is not a temporary storage location for messages. Instead, it provides bidirectional, unbuffered connections across network boundaries such as firewalls. Use a relay when you want direct communications between components as if they were located on the same network segment but separated by network security devices.

## Benefits of queues

Queue infrastructures can support many advanced features that make them very useful in the following ways.

***Message delivery guarantees***

Queuing systems usually guarantee delivery of each message in the queue to a destination component. However, these guarantees can take different approaches:

* **At-Least-Once Delivery**: In this approach, each message is guaranteed to be delivered to at least one of the components that retrieve messages from the queue. Note, however, that in certain circumstances, it is possible that the same message may be delivered more than once. If one instance takes a long time to process the message, and a time-out expires, the message may be sent to the other instance as well. Your web app code should be designed with this possibility in mind.
* **At-Most-Once Delivery**: In this approach, each message is not guaranteed to be delivered, and there is a very small chance that it may not arrive. However, unlike At-Least-Once delivery, there is no chance that the message will be delivered twice. This is sometimes referred to as "automatic duplicate detection".
* **First-In-First-Out** (FIFO): In most messaging systems, messages usually leave the queue in the same order in which they were added, but you should consider whether this order is guaranteed. If your distributed application requires that messages are processed in precisely the correct order, you must choose a queue system that includes a FIFO guarantee.

***Transactional support***

Some closely related groups of messages may cause problems when delivery fails for one message in the group. You can avoid these kinds of problems by grouping messages into a transaction. Message transactions succeed or fail as a single unit.

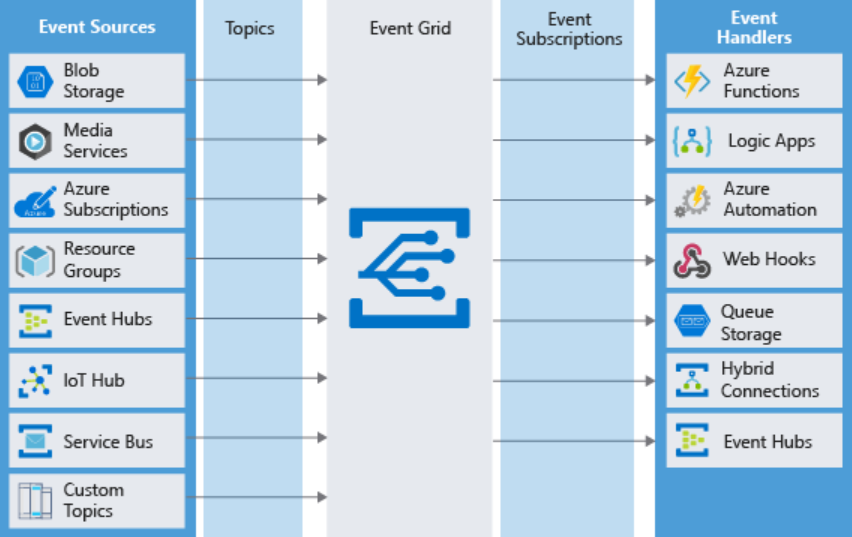
## Code with topics and queues

Even if you use topics or queues, in Visual Studio, you will use the **Microsoft.Azure.ServiceBus NuGet** package, configure connection strings, and asynchronous programming patterns. For it you'll use the **TopicClient** or the **QueueClient** class to send messages and the **SubscriptionClient** class to receive messages.

# Azure Event Grid

Azure Event Grid is a fully-managed event routing service running on top of Azure Service Fabric. Event Grid distributes events from different sources. Event Grid supports most Azure services as a publisher or subscriber and can be used with third-party services. It provides a dynamically scalable, low-cost, messaging system that allows publishers to notify subscribers about a status change.

The following illustration shows an Azure Event Grid positioned between multiple event sources and multiple event handlers. The event sources send events to the Event Grid and the Event Grid forwards relevant events to the subscribers. Event Grid use topics to decide which events to send to which handlers. Events sources tag each event with one or more topics, and event handlers subscribe to the topics they are interested in.



## When should use Event Grid

Use Event Grid when you need these features:

* **Simplicity**: It is straightforward to connect sources to subscribers in Event Grid.
* **Advanced filtering**: Subscriptions have close control over the events they receive from a topic.
* **Fan-out**: You can subscribe to an unlimited number of endpoints to the same events and topics.
* **Reliability**: Event Grid retries event delivery for up to 24 hours for each subscription.
* **Pay-per-event**: Pay only for the number of events that you transmit.

## Event

Events are the data messages passing through Event Grid that describe what has taken place. Each event is self-contained, can be up to 64 KB, and contains several pieces of information based on a schema defined by Event Grid:

|  |  |
| --- | --- |
| **Field** | **Description** |
| topic | The full resource path to the event source. Event Grid provides this value. |
| subject | Publisher-defined path to the event subject. |
| id | The unique identifier for event. |
| eventType | One of the registered event types for this event source. This is a value you can create filters against, e.g. CustomerCreated, BlobDeleted, HttpRequestReceived, etc. |
| eventTime | The time the event was generated based on the provider's UTC time. |
| data | Specific information that is relevant to the type of event. For example, an event about a new file being created in Azure Storage has details about the file, such as the lastTimeModified value. Or, an Event Hubs event has the URL of the Capture file. This field is optional. |
| 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. Event Grid provides this value. |

## Event source

Event sources are responsible for sending events to Event Grid. Each event source is related to one or more event types.

Azure Event Hub has the concept of an event publisher which is often confused with the event source. A publisher to Event Hub is the user or organization that decides to send events to Event Grid.

***Types of event sources***

Events can be generated by the following Azure resource types:

* **Azure Subscriptions and Resource Groups**: Subscriptions and resource groups generate events related to management operations in Azure. For example, when a user creates a virtual machine, this source generates an event.
* **Container registry**: The Azure Container Registry service generates events when images in the registry are added, removed, or changed.
* **Event Hub**: Event Hub can be used to process and store events from a variety of data sources - typically logging or telemetry related. Event Hub can generate events to Event Grid when files are captured.
* **Service Bus**: Service bus can generate events to Event Grid when there are active messages with no active listeners.
* **Storage accounts**: Storage accounts can generate events when users add blobs, files, table entries, or queue messages. You can use both blob accounts and General-purpose V2 accounts as event sources.
* **Media Services**: Media Services hosts video and audio media and provides advanced management features for media files. Media Services can generate events when an encoding job is started or completed on a video file.
* **Azure IoT Hub**: IoT Hub communicates with and gathers telemetry from IoT devices. It can generate events whenever such communications arrive.
* **Custom events**: Custom events can be generated using the REST API, or with the Azure SDK on Java, GO, .NET, Node, Python, and Ruby. For example, you could create a custom event in the Web Apps feature of Azure App Service. This can happen in the worker role when it picks up a message from a storage queue.

## Event topic

Event topics categorize events into groups. Topics are represented by a public endpoint and are where the event source sends events to. When designing your application, you can decide how many topics to create. Larger solutions will create a custom topic for each category of related events, while smaller solutions might send all events to a single topic.

## Event subscription

Event Subscriptions define which events on a topic an event handler wants to receive. A subscription can also filter events by their type or subject, so you can ensure an event handler only receives relevant events.

## Event handler

An event handler (sometimes referred to as an event "subscriber") is any component (application or resource) that can receive events from Event Grid.

***Types of event handlers***

The following object types in Azure can receive and handle events from Event Grid:

* **Azure Functions**: Custom code that runs in Azure, without the need for explicit configuration of a host virtual server or container. Use an Azure function as an event handler when you want to code a custom response to the event.
* **Webhooks**: A webhook is a web API that implements a push architecture.
* **Azure Logic Apps**: An Azure logic app hosts a business process as a workflow.
* **Microsoft Flow**: Flow also hosts workflows, but it is easier for non-technical staff to use.

# Azure Event Hubs

Event Hubs is an intermediary for the publish-subscribe communication pattern. Unlike Event Grid, however, it is optimized for extremely high throughput, a large number of publishers, security, and resiliency. For it is necessary to have first an Event Hubs Namespace. When creating a new Event Hub, there are several mandatory parameters.

* **Event Hub name**: Event Hub name that is unique within your subscription and:
  + Is between 1 and 50 characters long
  + Contains only letters, numbers, periods, hyphens, and underscores
  + Starts and ends with a letter or number
* **Partition Count**: The number of partitions required in an Event Hub (between 2 and 32). The partition count should be directly related to the expected number of concurrent consumers and can't be changed after the hub has been created. The partition separates the message stream so that consumer or receiver applications only need to read a specific subset of the data stream. If not defined, this value defaults to 4.
* **Message Retention**: The number of days (between 1 and 7) that messages will remain available if the data stream needs to be replayed for any reason. If not defined, this value defaults to 7.

## When should use Event Hub

Choose Event Hubs if:

* You need to support authenticating a large number of publishers.
* You need to save a stream of events to Data Lake or Blob storage.
* You need aggregation or analytics on your event stream.
* You need reliable messaging or resiliency.

## Partitions

As Event Hubs receives communications, it divides them into partitions. Partitions are buffers into which the communications are saved. Because of the event buffers, events are not completely ephemeral, and an event isn't missed just because a subscriber is busy or even offline. The subscriber can always use the buffer to "catch up."

## Capture

Event Hubs can send all your events immediately to Azure Data Lake or Azure Blob storage for inexpensive, permanent persistence.

## Authentication

All publishers are authenticated and issued a token. This means Event Hubs can accept events from external devices and mobile apps, without worrying that fraudulent data from pranksters could ruin our analysis.

## Application requirements

To configure an application to send messages to an Event Hub, you must provide the following information, so that the application can create connection credentials:

* Event Hub namespace name
* Event Hub name
* Shared access policy name
* Primary shared access key

To configure an application to receive messages from an Event Hub, provide the following information, so that the application can create connection credentials:

* Event Hub namespace name
* Event Hub name
* Shared access policy name
* Primary shared access key
* Storage account name
* Storage account connection string
* Storage account container name

## Performance evaluation

Evaluate your Event Hub by testing that your Event Hub is processing data as expected. The metrics available in the Event Hubs allow you to ensure that it's working fine. The Azure portal provides message counts and other metrics that you can use as a health check for your Event Hubs.