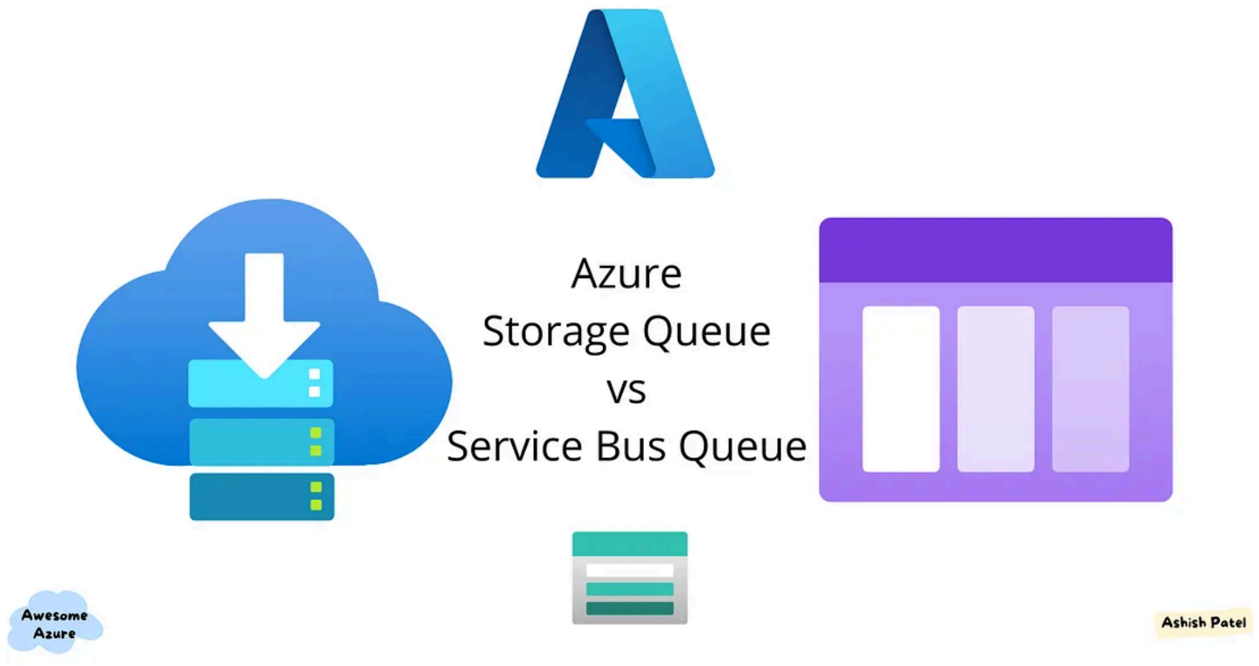
Azure Queue Vs Azure Service Bus Queue



One of the benefits of cloud computing is that it allows you to scale your computing resources up or down depending on specific workloads. To take advantage of this feature, software solutions need to be architected so that the different components are independent and solely responsible for the tasks assigned to them. This loose coupling prevents unnecessary bottlenecks that could potentially slow down the entire solution.

One common way to decouple solutions is to use messaging queues, recognized as an enterprise architecture pattern that facilitates the integration of disconnected components. In the cloud, messaging queues have become the canonical example of how to properly segregate functional layers that perform their job as effectively as possible.

In Azure, the messaging architecture pattern can be implemented by using two ways: Azure Queues and Service Bus Queues.

Performance

The most significant difference between Azure queues and Service Bus queues is their average latency: **10 milliseconds for the former and 100 milliseconds for the latter**, assuming that the request is generated from a hosted service located in the same location as the storage account. Both types support up to 2,000 messages per second.

From a developer's point of view, this means that when traffic grows, the number of Service Bus queues has to increase more rapidly than when using Azure queues.

Capacity

Messaging pattern capacity is limited/defined by three factors: message size, storage supported per queue and the amount of time that messages are kept alive (or TTL, for time-to-live).

When it comes to Azure queues, each message can be **up to 64 KB, can hold up to 1 TB of data** and can store messages for up to 7 days. Service Bus queues support messages **up to 256 KB and are limited to 5 GB of data**, but their TTL for messages is unrestricted.

These limits are important when it comes to processing messages in both frameworks, particularly when dealing with situations in which the application crashes after processing the message but before the message is removed from the queue.

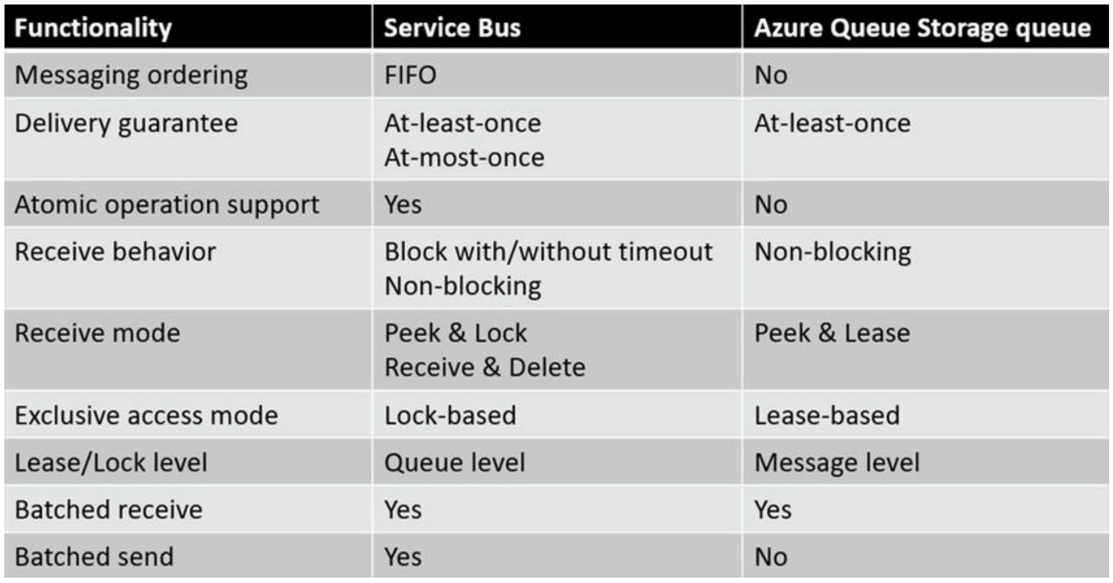
Security

Authentication for storage queues is limited to the use of a security token, which is shared by other components (including tables and binary large objects, or BLOBs). For this reason, the application needs to handle all the security when a more sophisticated mechanism is required.

Getting access to the Azure queue is as simple as providing the **storage account name and the token associated** with it.

For Service Bus queues, **Azure Access Control Service** (ACS) is used as the primary authentication mechanism, supporting multiple predefined identity providers as well as custom ones.

So, Azure Queues or Service Bus Queues?



Functionality	Service Bus	Azure Queue Storage queue
Messaging ordering	FIFO	No
Delivery guarantee	At-least-once At-most-once	At-least-once
Atomic operation support	Yes	No
Receive behavior	Block with/without timeout Non-blocking	Non-blocking
Receive mode	Peek & Lock Receive & Delete	Peek & Lease
Exclusive access mode	Lock-based	Lease-based
Lease/Lock level	Queue level	Message level
Batched receive	Yes	Yes
Batched send	Yes	No

Azure Service Bus Queues are best for **complex, high-value enterprise integration scenarios** that require advanced features like **guaranteed message ordering** (FIFO), **transactions, or pub/sub capabilities**. So, they are the preferred choice.

Azure Queues are best for **simple, high-volume, high-throughput, decoupled application** scenarios that need basic task queuing and are the most **cost-effective option for large-scale simple queuing**.