Azure Queue Storage vs Azure Service Bus

Aspect	Azure Queue Storage	Azure Service Bus
Definition	Simple message queue built on Azure Storage for basic asynchronous communication.	Enterprise grade messaging system for complex, reliable communication between distributed apps.
Architecture	Producer → Queue → Consumer (Single sender & receiver)	Producer → Topic → Multiple Subscribers (Supports pub/sub model)
RealTime Example	Ecommerce site queues image processing tasks after upload.	Banking system sends transaction event to multiple services (balance update, fraud check, notifications).
Use When	• Simple background tasks • One sender & one receiver • Cost sensitive apps • No need for ordering or retries	• Complex workflows • Multiple consumers • Need for ordering, retries, dead lettering • Enterprise grade reliability
Message Size Limit	Up to 64 KB	Up to 1 MB (Standard), 100 MB (Premium)
Ordering	Not guaranteed	Guaranteed using sessions
Delivery Guarantee	At least once	At least once, exactly once (with transactions)
Dead lettering	Not supported	Supported
Duplicate Detection	Not supported	Supported
Transactions	Not supported	Supported
Peek lock (Read without delete)	Not supported	Supported
Auto-forwarding	Not supported	Supported

Topics & Subscriptions	Not available	Available (pub/sub model)
Security	Shared Access Signature (SAS)	SAS, Role Based Access Control (RBAC), Managed Identity
Protocol	HTTP/HTTPS	AMQP, HTTPS
Monitoring	Basic via Azure Monitor	Advanced metrics and diagnostics
Scalability	Scales well for simple workloads	Scales for high throughput, complex systems
Cost	Low	Higher (especially Premium tier)
Best For	Lightweight apps, background jobs, simple task queues	Financial systems, logistics, healthcare, enterprise apps

Azure Queue Storage, Use When?

1. You need a simple task queue

- You just want to queue tasks like sending emails, resizing images, or processing files.
- No complex routing or multiple receivers.

2. Cost is a concern

- Queue Storage is cheaper and ideal for small-scale or budget-sensitive apps.

3. You don't need message ordering

- Messages may not be processed in the exact order they were sent, and that's okay.

4. You don't need advanced features

- No need for duplicate detection, transactions, or dead lettering.

5. You have one sender and one receiver

- Simple producer-consumer model.

Azure Service Bus, Use When?

1. You need reliable and ordered delivery

- Critical for financial systems, inventory updates, or anything where order matters.

2. You have multiple receivers

- Use **topics and subscriptions** to broadcast messages to many services.

3. You need advanced features

- Supports:
    - **Dead-lettering** (for failed messages)
    - **Duplicate detection**
    - **Transactions**
    - **Sessions** (for ordered processing)

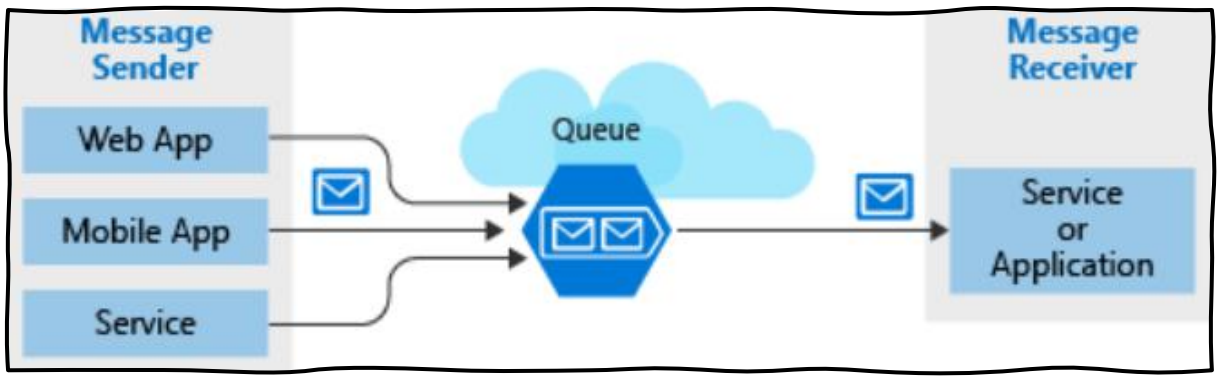
4. You want to track message delivery

- You can peek-lock messages, retry on failure, and ensure they're processed exactly once.

5. You're building an enterprise-grade system

- Ideal for banking, logistics, healthcare, or any large-scale distributed system.

Azure Queue Storage Architecture – Simple Task Queue



Components:

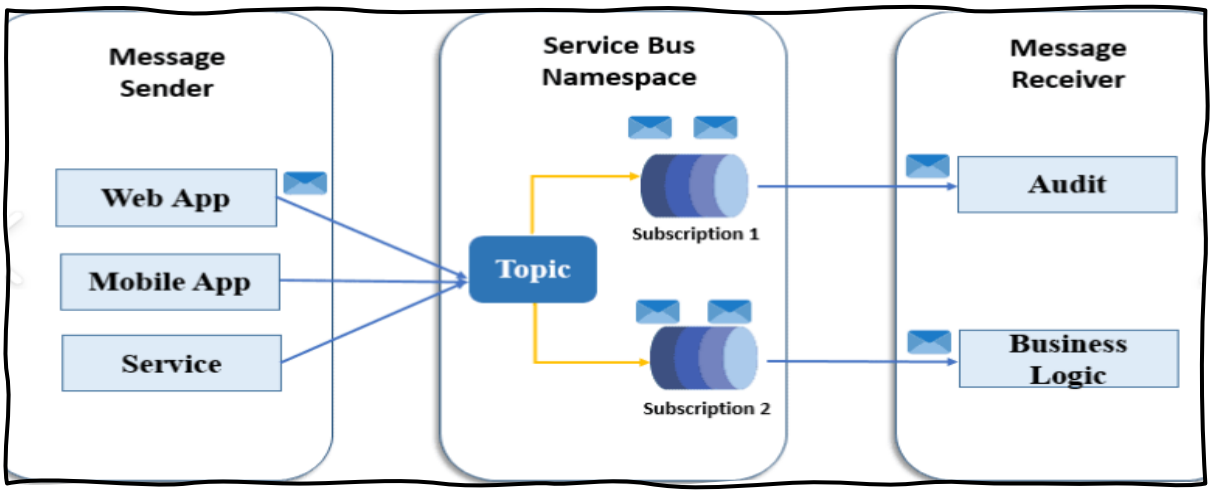
1. **Producer App** – Sends messages to the queue.
2. **Azure Queue Storage** – Stores messages.

3. **Consumer App** – Reads and processes messages.

Example Flow (Image Processing):

- Customer uploads image →
  Producer app sends message: "Resize image for product 123" →
  Message goes into Azure Queue →
  Consumer app picks message →
  Resizes image →
  Saves result.

Azure Service Bus Architecture – Enterprise Messaging



Components:

1. **Producer App** – Sends messages to a **Topic** or **Queue**.
2. **Azure Service Bus** – Manages messages, supports routing, sessions, retries.
3. **Subscribers/Consumers** – Multiple apps/services that receive messages.

Example Flow (Bank Transaction):

- Customer makes transaction →
  Producer app sends message to **Service Bus Topic** →
  Multiple subscribers:
  - One updates balance
  - One checks fraud
  - One sends SMS
  - One logs transaction