
Azure

Azure Logic Apps

Grow Data Skills

What is Azure Logic Apps?

- Azure Logic Apps is a cloud-based service that helps you automate workflows and integrate apps, data, services, and systems.
- It offers a visual designer with a drag-and-drop interface, enabling both developers and non-developers to create complex workflows with minimal coding.
- Designed to support scalable, event-driven, and serverless applications with real-time processing.

Key Features:

- ***Pre-Built Connectors:*** Easily connect to over 300 built-in connectors, including Microsoft services (Office 365, Dynamics 365, SharePoint) and third-party services (Salesforce, Twitter, SAP).
- ***Triggers and Actions:*** Start workflows with events (triggers) like file uploads or HTTP requests and execute actions like sending emails, inserting records, or calling APIs.
- ***Scalability and Reliability:*** Automatically scales based on demand, with built-in fault tolerance and retry mechanisms.
- ***Enterprise Integration:*** Support for industry-standard protocols and formats like XML, JSON, EDI, and AS2, making it ideal for B2B integration scenarios.

Use Cases & How Logic Apps Work

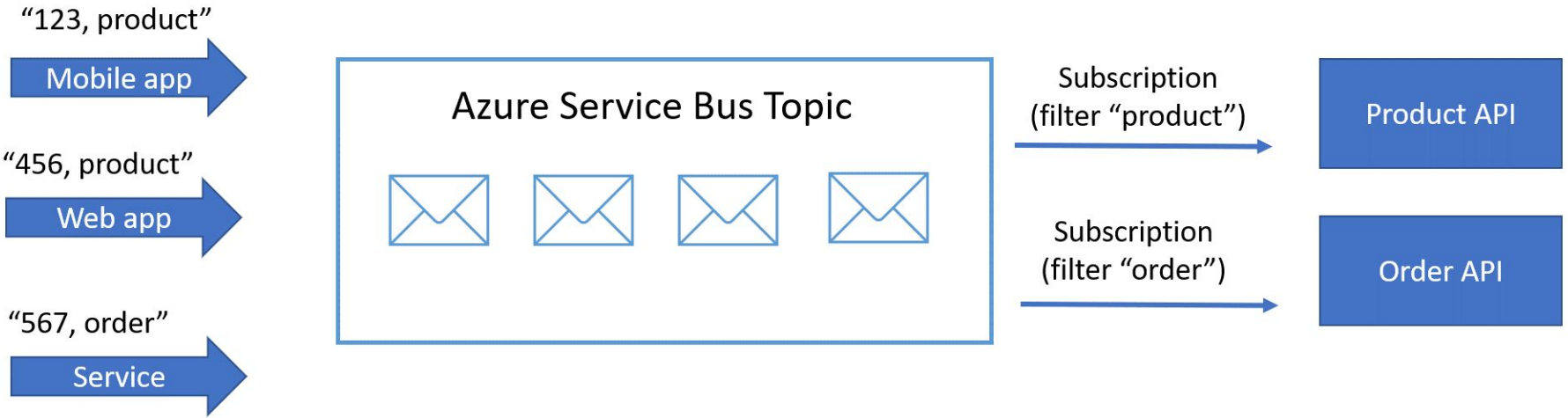
Grow Data Skills

Common Use Cases:

- ***Business Process Automation:*** Streamline approval processes, employee onboarding, and document management workflows.
- ***Data Synchronization:*** Sync data across multiple systems like CRM, ERP, and cloud platforms (e.g., sync data between SQL Server and SharePoint).
- ***Event-Driven Processing:*** Trigger workflows based on real-time events such as file uploads in Azure Blob Storage, database updates, or IoT device signals.
- ***Monitoring and Notifications:*** Automatically send notifications when specific events occur, like monitoring website performance and sending alerts when issues are detected.

How Logic Apps Work:

- ***Trigger***: The workflow starts with a trigger. This could be an event like receiving an HTTP request, detecting a new file in storage, or a scheduled time-based trigger.
- ***Actions***: After the trigger, a series of actions are executed. Actions can include tasks like sending an email, processing data, invoking APIs, or updating records in a database.
- ***Conditions and Loops:*** Incorporate logic into your workflows with conditions (if-else branching), loops (for-each), and switches for handling complex decision-making processes.
- ***Connectors***: Logic Apps offer built-in connectors that provide seamless integration with cloud services, on-premises systems, and third-party applications.


"123, product"
Mobile app
"456, product"
Web app
"567, order"
Service
Azure Service Bus Topic
Subscription
(filter "product")
Subscription
(filter "order")
Product API
Order API

Azure Service Bus (Topic)

Grow Data Skills

- Azure Service Bus is a fully managed enterprise message broker that allows reliable message queuing and communication between different services and applications, even across hybrid cloud environments.
- **Service Bus Topics** are ideal for publish-subscribe (pub-sub) messaging scenarios where multiple subscribers need to receive a message broadcasted by a single publisher.

Key Features of Service Bus Topics:

- ***Publish-Subscribe Messaging Model:*** Enables one-to-many communication by allowing multiple subscribers to receive the same message.
- ***Message Filtering and Routing:*** Define rules to filter and route messages to specific subscriptions based on properties.
- ***Advanced Messaging Features:*** Support for message sessions (ordered delivery), dead-lettering (handling undeliverable messages), message deferral, and time-to-live (TTL).
- ***Scalability and Reliability:*** Built-in support for load balancing, partitioning, and reliable delivery guarantees with features like duplicate detection.

Azure Service Bus (Topic)

Grow Data Skills

Use Cases & How Service Bus Topics Work

- **<u>Common Use Cases:</u>**
    - ***Decoupling Microservices:*** Enable loosely coupled microservices communication where one service publishes a message that multiple services can subscribe to.
    - ***Event Distribution:*** Broadcast events like order placement, payment confirmation, or sensor data where multiple downstream systems need to react (e.g., logging, analytics, and notifications).
    - ***Load Balancing:*** Distribute workload among multiple consumers in a balanced manner by leveraging subscriptions and rules.
    - ***Multi-tenant Applications:*** Route messages based on tenant-specific rules, allowing each tenant to have isolated processing.

- How Service Bus Topics Work:
    - ***Topic Creation***: Define a Service Bus Topic where messages are published by a sender.
    - ***Subscriptions***: Each Topic can have multiple subscriptions. Subscriptions act like virtual queues that receive copies of messages sent to the topic.
    - ***Message Publishing***: A publisher sends messages to the topic, and all active subscriptions receive copies of the message.
    - ***Rules and Filters***: Subscriptions can be configured with rules and filters to control which messages they receive, allowing for targeted message delivery.
    - ***Dead-Letter Queue (DLQ):*** Undeliverable or expired messages are moved to a dead-letter queue for further inspection and manual intervention.

Azure Service Bus (Queue)

Grow Data Skills

What is Azure Service Bus Queue?

- Azure Service Bus Queue is a fully managed enterprise message broker that facilitates communication between distributed applications and services by enabling message queuing.
- It follows the FIFO (First In, First Out) model, ensuring messages are processed in the order they are received.
- Unlike topics, which allow for publish-subscribe scenarios, queues are designed for point-to-point messaging, where each message is processed by a single receiver.

Key Features:

- ***Message Delivery Guarantee:*** Supports guaranteed message delivery using Peek-Lock and Receive-Delete modes.
- ***Dead-Letter Queues (DLQ):*** Provides a built-in mechanism to handle messages that can't be processed, allowing for more robust error handling.
- ***Scheduled Delivery:*** Allows you to delay the delivery of messages until a specified time.
- ***Message Sessions:*** Enables ordered message processing for scenarios requiring sequence guarantees.
- ***Advanced Features:*** Includes duplicate detection, message deferral, and transaction support.

Azure Service Bus (Queue)

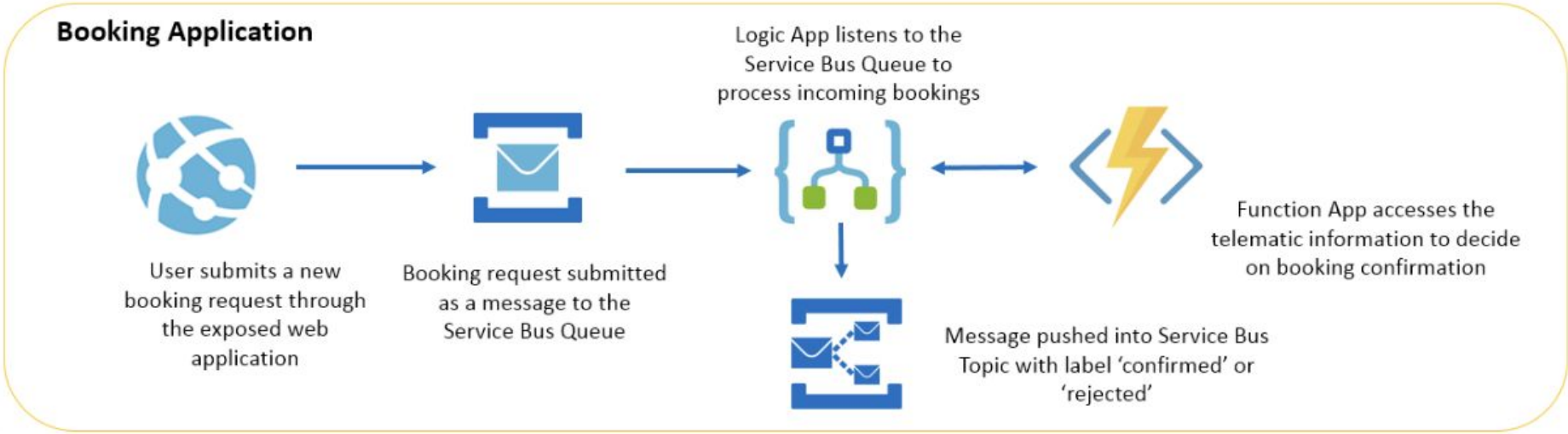
Grow Data Skills

Common Use Cases:

- ***Decoupling Microservices:*** Facilitate asynchronous communication between microservices where services don't need to be aware of each other's availability.
- ***Load Leveling***: Queue messages when the system is under heavy load, allowing downstream processing services to handle them at their own pace.
- ***Order Processing***: Manage tasks like order fulfillment where messages need to be processed sequentially.
- ***Resilient Processing***: Improve fault tolerance by using dead-letter queues to capture and retry failed messages.

How Azure Service Bus Queues Work:

- ***Message Producer:*** An application or service sends a message to the queue.
- ***Queue***: The message waits in the queue until it is retrieved and processed by a consumer.
- ***Message Consumer:*** The consumer receives the message, processes it, and optionally completes or abandons it.
- ***Receive Modes:***
    - Peek-Lock: The message is locked while being processed and is only removed when processing is successfully completed.
    - Receive-Delete: The message is immediately deleted once it is received, making this mode faster but less reliable in case of failure.
- ***Dead-Letter Queue (DLQ):*** Messages that cannot be processed (due to errors or expiration) are moved to a dead-letter queue for further investigation.


Booking Application
Logic App listens to the Service Bus Queue to process incoming bookings
User submits a new booking request through the exposed web application
Booking request submitted as a message to the Service Bus Queue
Function App accesses the telematic information to decide on booking confirmation
Message pushed into Service Bus Topic with label 'confirmed' or 'rejected'