St10448749
Joshua Gombele

## Student Information

Student Number: ST10448749

Module Code: CLDV6212

Part: 2

St10448749
Joshua Gombele

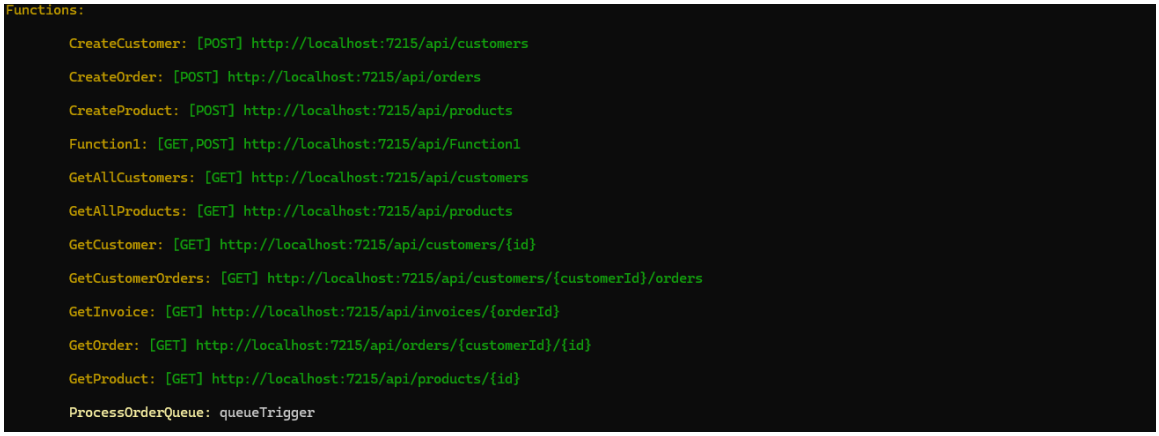
## Application Details and Screenshots

Deployed Application URL: https://abcretailersjoshua-g4btbkdcdnebbddq.southafricanorth-01.azurewebsites.net

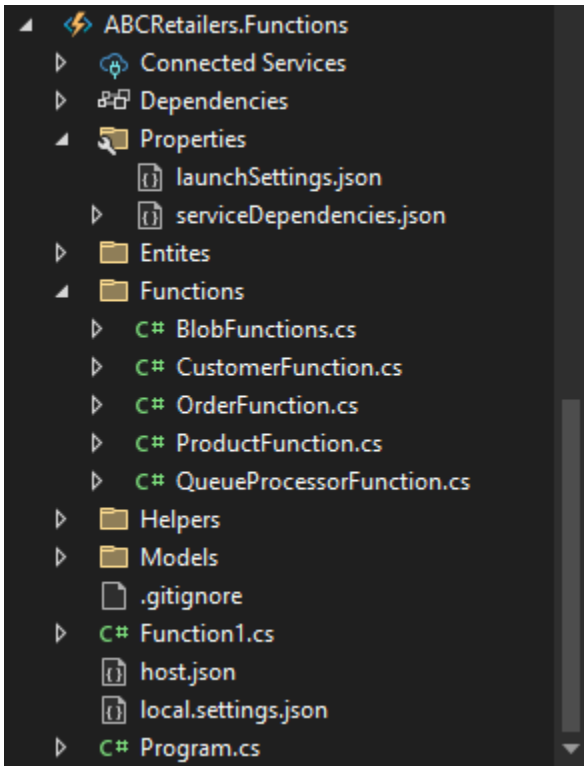
GitHub Repository Link: https://github.com/BokosaJG/cldv6212poepart2.git

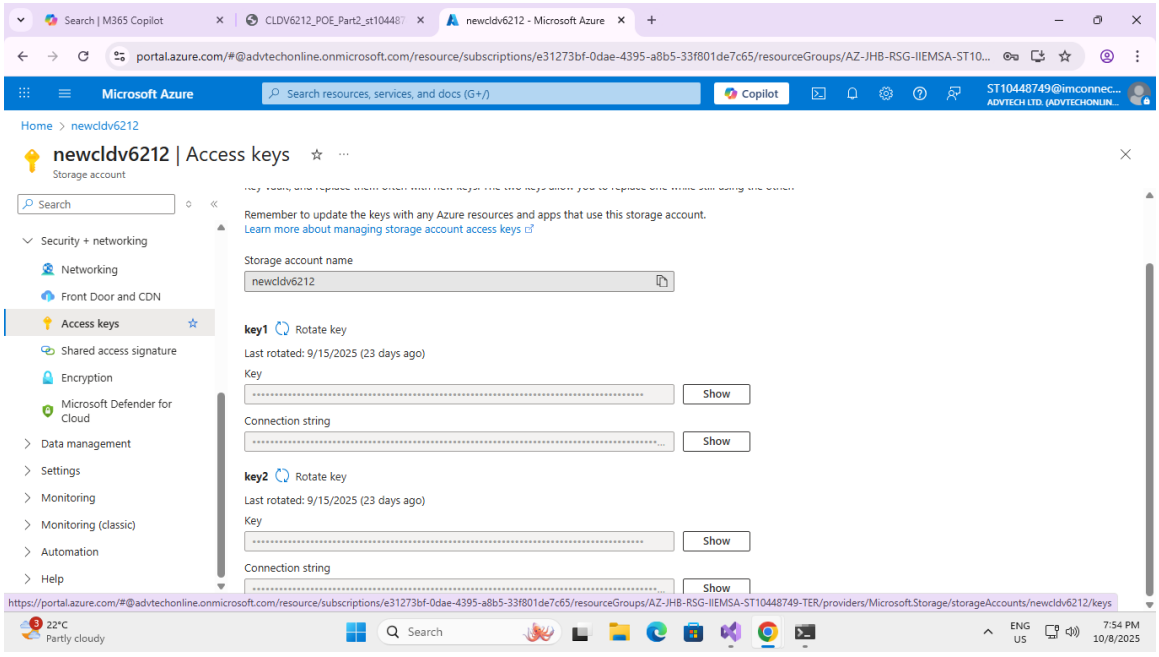
Youtube Link: https://youtu.be/3igjLBNPLPc?si=3oWy2jvPj0oOW1zh

All the screenshots required

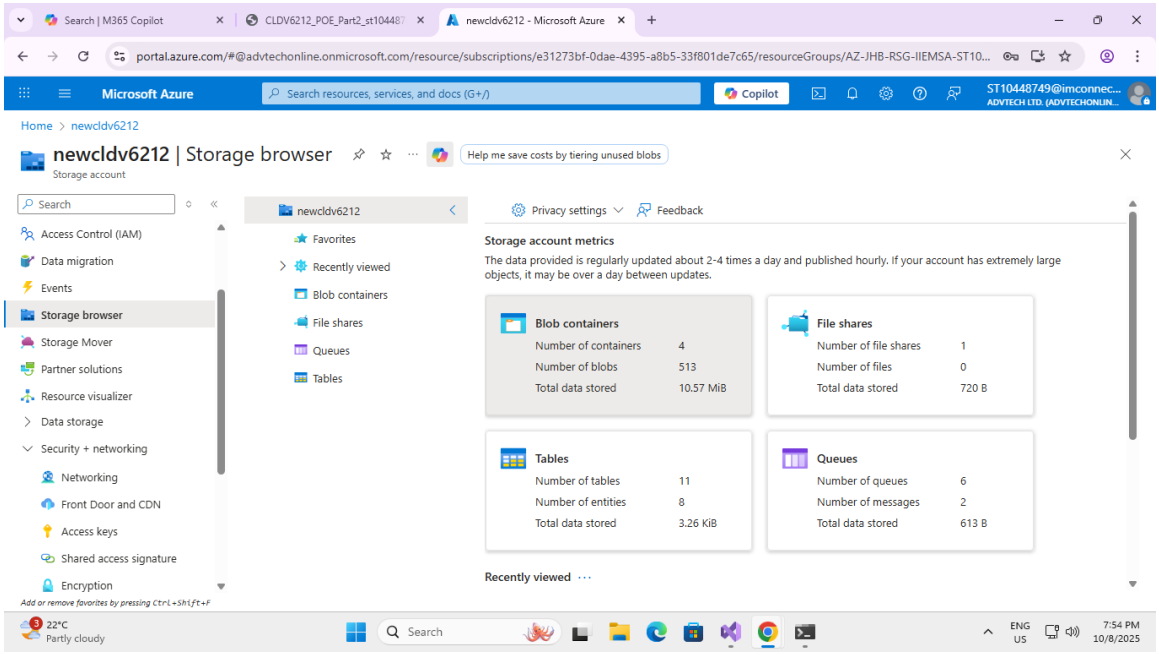


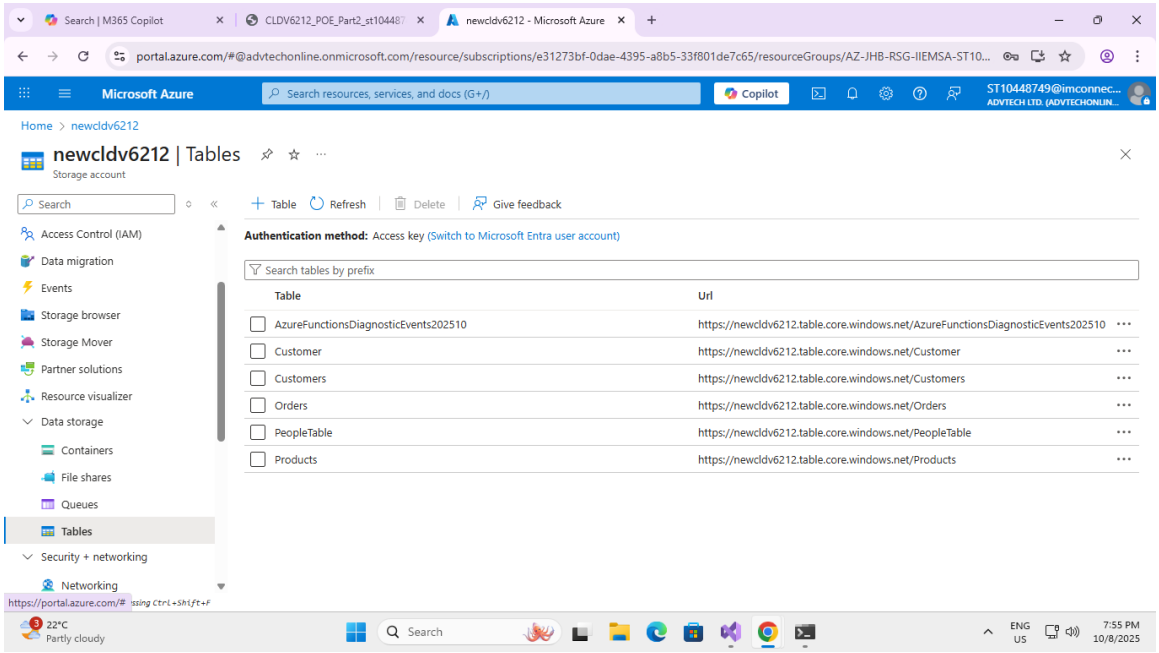

St10448749
Joshua Gombele



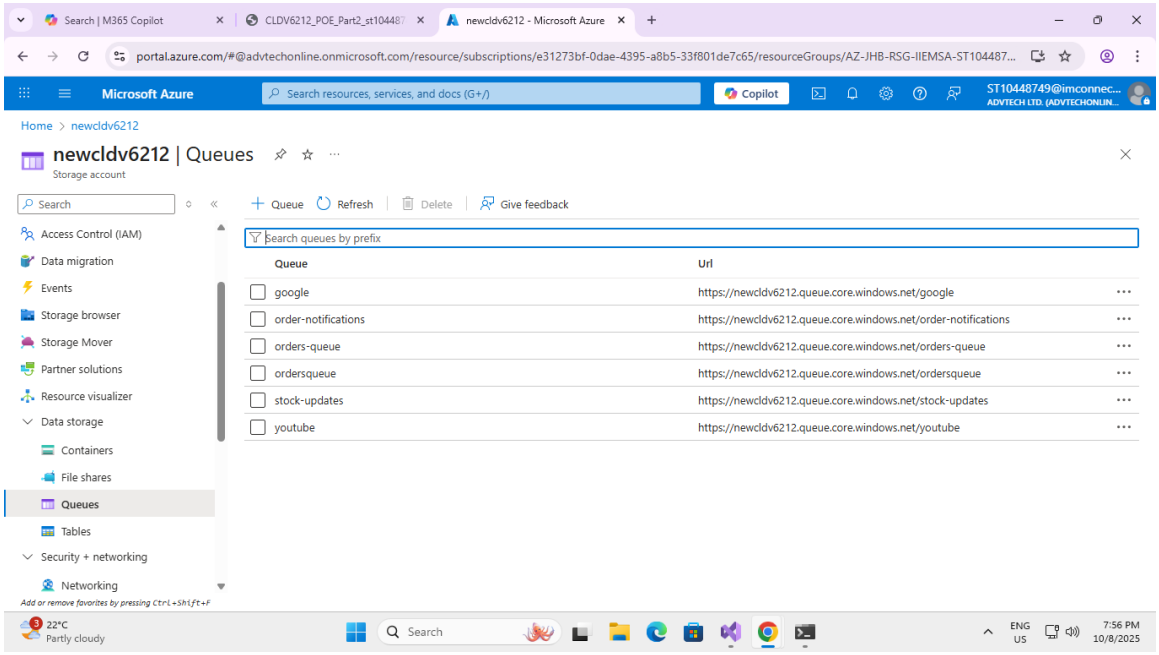



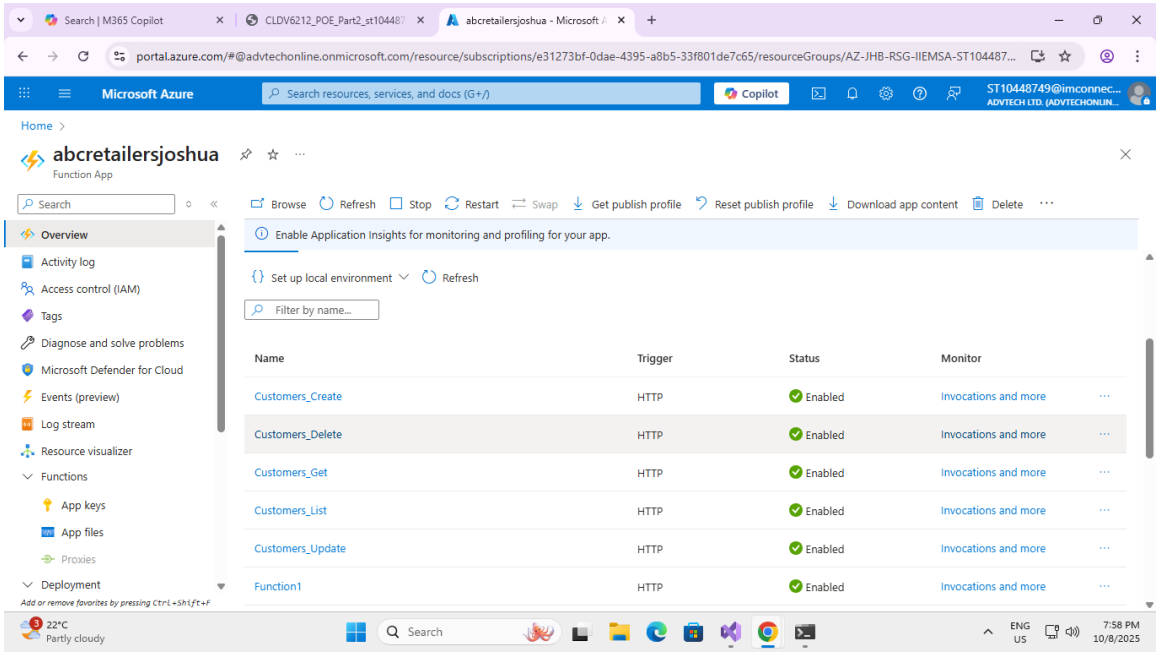
St10448749
Joshua Gombele



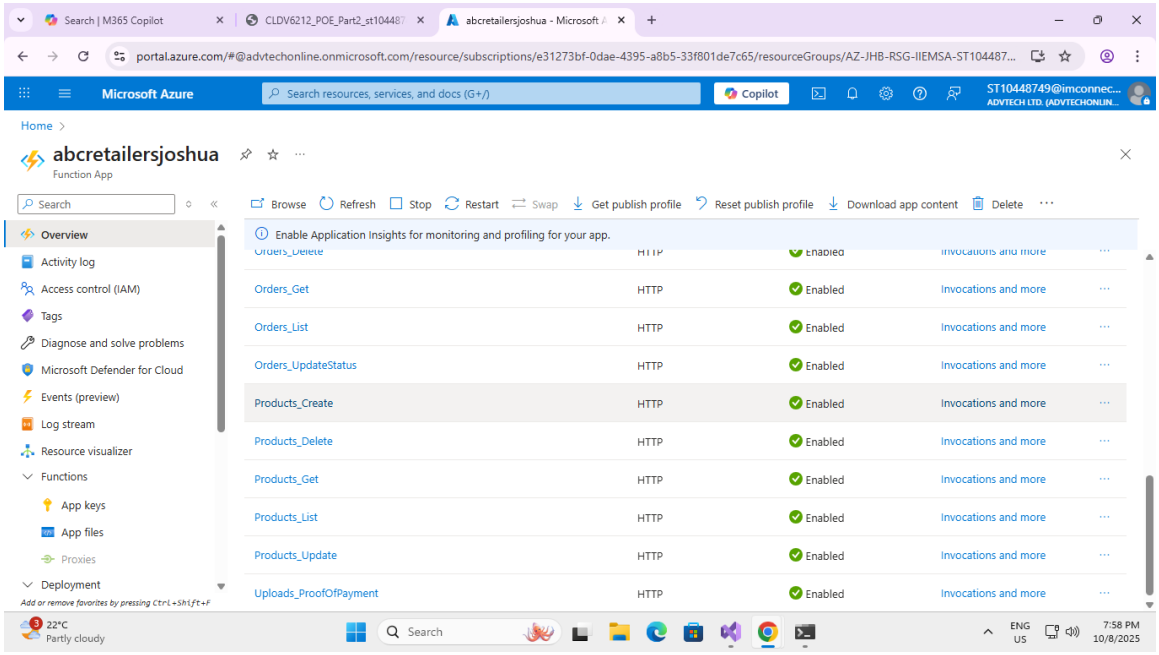



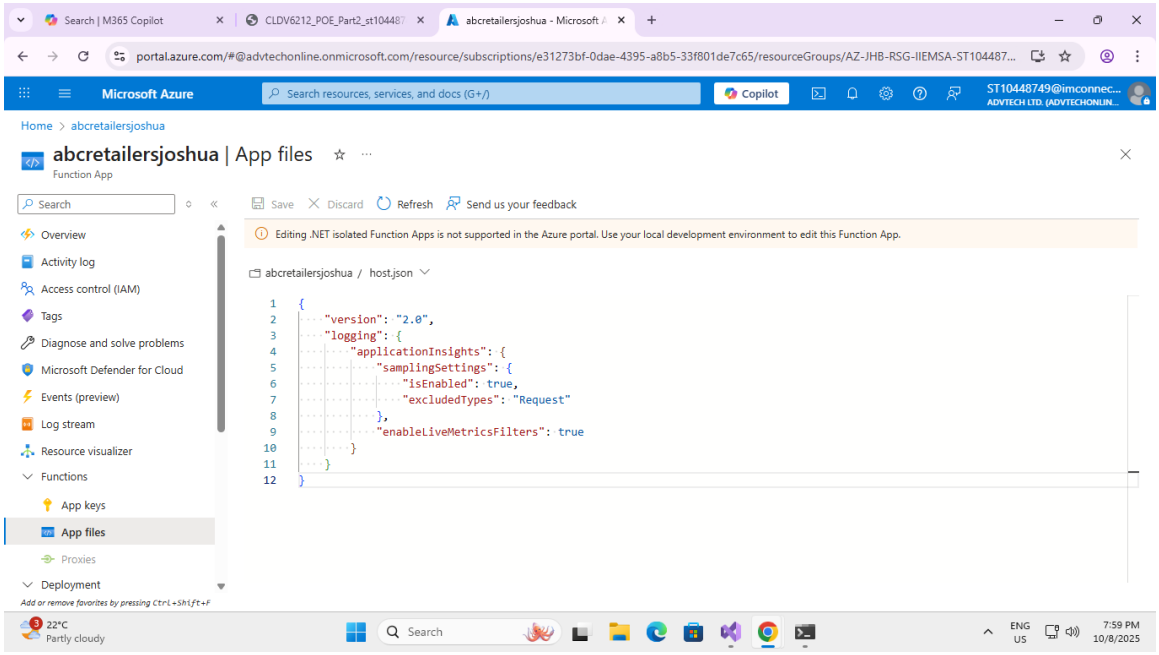
St10448749
Joshua Gombele



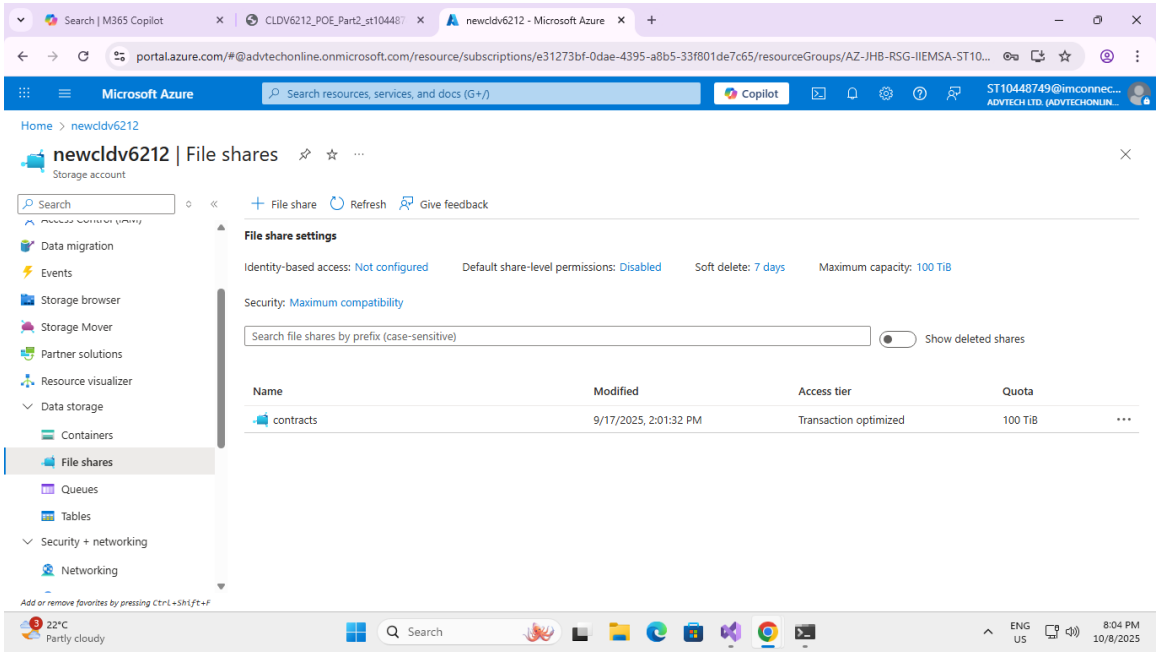



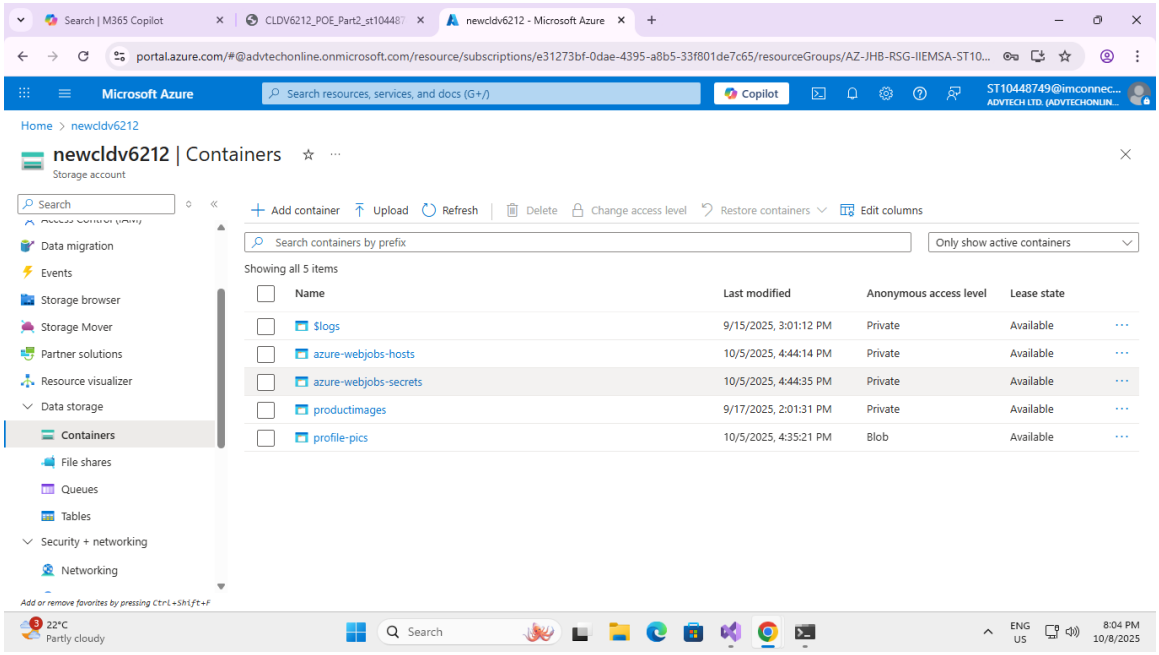
St10448749
Joshua Gombele



St10448749
Joshua Gombele



St10448749
Joshua Gombele

# Section B: Improving Customer Experience with Azure Services

## Azure Event Hubs: The High-Scale Data Ingestion Pipeline

Azure Event Hubs is designed as a "big data streaming platform" and an event ingestion service. Think of it as the superhighway for data, built to handle a massive torrent of information from a multitude of sources simultaneously.

**Detailed Description & Core Architecture**

At its heart, Event Hubs is a highly scalable **publish-subscribe** event ingestor. Its architecture is optimized for throughput and scale, not for complex message routing or guaranteed ordered processing per individual entity.

- **Event Producers (Publishers):** Any device or application that sends data to an Event Hub. This could be IoT devices, websites, mobile apps, or other services.
- **Event Hub:** The central endpoint. It receives the stream of events. An Event Hub is divided into **partitions**.
- **Partitions:** This is a key concept. Each partition is an ordered sequence of events. When an event is sent, it can be assigned a **partition key** to group related events (e.g., all events from a specific device ID) into the same partition, ensuring order for that group. Events are then read in the order they were received *within that partition*.
- **Consumer Groups:** These are parallel, independent views of the entire event stream. Each consumer group allows a set of consumers (applications) to have its own separate read position in the stream. This enables multiple downstream systems (e.g., one for hot-path analytics, another for cold-path archival) to process the same stream of events independently without interfering with each other.

**In-Depth Mechanism: How It Works**

1. **Massive Ingestion:** Event Hubs is built on a model of temporal retention. Events are sent to the Event Hub and are stored for a configurable retention period (1 to 7 days, default is 1). It uses an **AMQP 1.0** protocol, optimized for sustained, high-volume connections.
2. **Partitioning for Parallelism:** The data stream is partitioned to allow for massive parallel consumption. While the overall stream is unordered, each partition guarantees order. This is a trade-off for immense scalability.

3. **The Event Processor Pattern:** Consumers don't typically read messages one-by-one. Instead, they use a checkpoint-based model (via the EventProcessorClient). A client leases a partition, reads events in batches from that partition, and periodically checkpoints its progress. If it fails, another client can take over the partition and resume from the last checkpoint.

4. **Integration with Azure Data Services:** Event Hubs is the natural starting point for the Azure big data ecosystem. It seamlessly integrates with:

    - **Azure Stream Analytics:** For real-time SQL-like queries on the streaming data.
    - **Azure Databricks / Spark:** For complex event processing (CEP) and machine learning on streams.
    - **Azure Data Lake / Azure Blob Storage:** For long-term archival of the event stream.

**Detailed Value to End Users & Business Impact**

- **Real-Time Analytics and Personalization:** By processing a live stream of user interactions (clicks, views, searches), businesses can build systems that provide **personalized recommendations** within milliseconds. This directly increases user engagement and conversion rates.

- **Faster Customer Feedback and Insights:** In scenarios like a live telemetry feed from a connected car or a manufacturing sensor, Event Hubs enables immediate anomaly detection. This allows for **proactive customer service** (e.g., alerting a driver of a potential issue) or optimizing production lines in real-time, leading to higher quality and customer satisfaction.

- **Scalability for Unpredictable Loads:** It can automatically scale to handle sudden, massive traffic spikes (e.g., during a product launch or a major sporting event) without any application changes, ensuring data is never lost.

- **Cost-Effective Data Ingestion:** By aggregating millions of small events into a single stream, it reduces the cost and complexity of connecting thousands of individual data sources directly to processing engines.

St10448749
Joshua Gombele

# Azure Service Bus

**Azure Service Bus: The Reliable Enterprise Messaging Backbone**

Azure Service Bus is a fully managed **Enterprise Message Broker**. It is designed for reliable, asynchronous communication between decoupled enterprise applications. Think of it as the highly secure, guaranteed postal service for your business-critical messages.

**Detailed Description & Core Architecture**

Service Bus provides several structured communication patterns beyond simple ingestion, focusing on delivery guarantees, transactions, and complex routing.

- **Queues:**
    - **Point-to-Point Communication:** A standard queue provides **First-In, First-Out (FIFO)** message delivery to a single consumer. Once a message is received and processed, it is deleted from the queue.
    - **Competing Consumers Pattern:** Multiple consumers can listen to the same queue, but only one will successfully process and remove a given message. This allows you to scale out your processing workload.
- **Topics and Subscriptions:**
    - **Publish-Subscribe Pattern:** A topic is similar to an Event Hub, but with a crucial difference: it has **subscriptions**. A message sent to a topic is copied to each subscription.
    - **Filtering and Routing:** Each subscription can have a **SQL filter rule** to receive only a subset of messages sent to the topic. For example, a "Order" topic could have subscriptions for "HighValueOrders," "EuropeanOrders," and "AllOrders," each with its own set of processing logic.

**In-Depth Mechanism: How It Works**

1. **Guaranteed, Asynchronous Delivery:** Service Bus is built for reliability. It provides:
    - **Durability:** Messages are stored transactionally and survive service restarts.
    - **Delivery Guarantees: At-Least-Once** semantics (you will not lose a message) and, with sessions, **Exactly-Once** processing.

- **Dead-Letter Queues (DLQ):** Messages that cannot be delivered or processed after a certain number of retries are moved to a separate sub-queue for manual inspection and corrective action.

2. **Sessions for FIFO and State:** This is a critical feature. **Message Sessions** guarantee strict FIFO ordering for a related set of messages (e.g., all actions for a specific customer order). They also allow storing application state alongside the message stream.

3. **Transactions:** Service Bus supports atomic transactions. You can send multiple messages to different queues/topics, or combine a message send/receive with a database update, and commit them as a single, all-or-nothing operation.

4. **Advanced Protocols:** Supports AMQP 1.0 (standard) and SOAP-based protocols, making it suitable for integrating with legacy on-premises systems like BizTalk Server.

**Detailed Value to End Users & Business Impact**

- **Ensures Order and Payment Reliability:** In an e-commerce system, when a customer places an order, an "OrderPlaced" message is sent to Service Bus. The system can guarantee this message is delivered and processed exactly once, even if the inventory service is temporarily down. This **builds trust** with customers, as they know their financial transactions are handled securely and reliably.

- **Improves System Resilience and Decoupling:** By using queues, if a backend service fails, messages simply wait in the queue until the service recovers. This prevents cascading failures and makes the overall system highly resilient and available.

- **Orchestrates Complex Business Workflows:** Using Topics with filtering, you can build sophisticated routing logic. For example, an "EmployeeHired" message can automatically trigger workflows in HR, IT (for account creation), and Facilities (for desk assignment) simultaneously and independently.

- **Integrates Hybrid Environments:** Its support for open standards and advanced security features (like Virtual Network integration) makes it the ideal backbone for connecting cloud-native applications with existing on-premises enterprise systems.

St10448749
Joshua Gombele

## References


- Microsoft. (2024). *What is Azure Event Hubs?*. [online] Azure.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about [Accessed 3 October 2025].
- Microsoft. (2024). *Azure Event Hubs - features and terminology*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features [Accessed 3 October 2025].
- Microsoft. (2024). *Consumer groups in Azure Event Hubs*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-features#consumer-groups [Accessed 3 October 2025].
- Microsoft. (2024). *Azure Event Hubs for Apache Kafka ecosystems*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/event-hubs/azure-event-hubs-kafka-overview [Accessed 3 October 2025].
- Microsoft. (2024). *Azure Event Hubs - Use Cases*. [online] Azure.microsoft.com. Available at: https://azure.microsoft.com/en-gb/products/event-hubs/ [Accessed 6 October 2025].
- Microsoft. (2024). *What is Azure Service Bus?*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview [Accessed 4 October 2025].
- Microsoft. (2024). *Service Bus queues, topics, and subscriptions*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions [Accessed 6 October 2025].
- Microsoft. (2024). *Message sessions in Azure Service Bus*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/service-bus-messaging/message-sessions [Accessed 6 October 2025].
- Microsoft. (2024). *Service Bus dead-letter queues*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dead-letter-queues [Accessed 6 October 2025].
- Microsoft. (2024). *Azure Service Bus - Use Cases*. [online] Azure.microsoft.com. Available at: https://azure.microsoft.com/en-gb/products/service-bus/ [Accessed 6 October 2025].