Introduction to Kafka

Jinish Bhardwaj
Software Architect, ASCIO
http://jinishbhardwaj.wordpress.com

Microsoft CERTIFIED Professional

Scrum Alliance CSM CERTIFIED

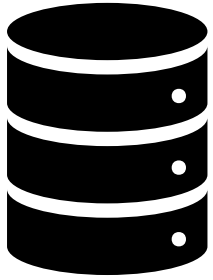
What will we cover

- What and Why of Kafka
  - Introduction to Confluent Kafka Dotnet Client
- Kafka architecture
  - Kafka vs MSMQ
  - Topics, Producers, Consumers, Messages
  - Message Delivery Semantics
- Use cases within Ascio
  - Looking at Ascio Order Processing
- Demo
  - Create a simple C# application in Visual Studio using Kafka Docker Image

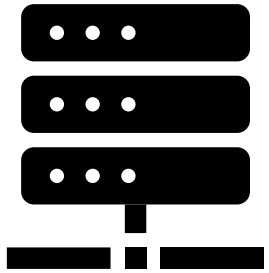
Kafka is more than just a message queue

Distributed Streaming Platform

Messaging system



Distributed data storage



Sequential data processing

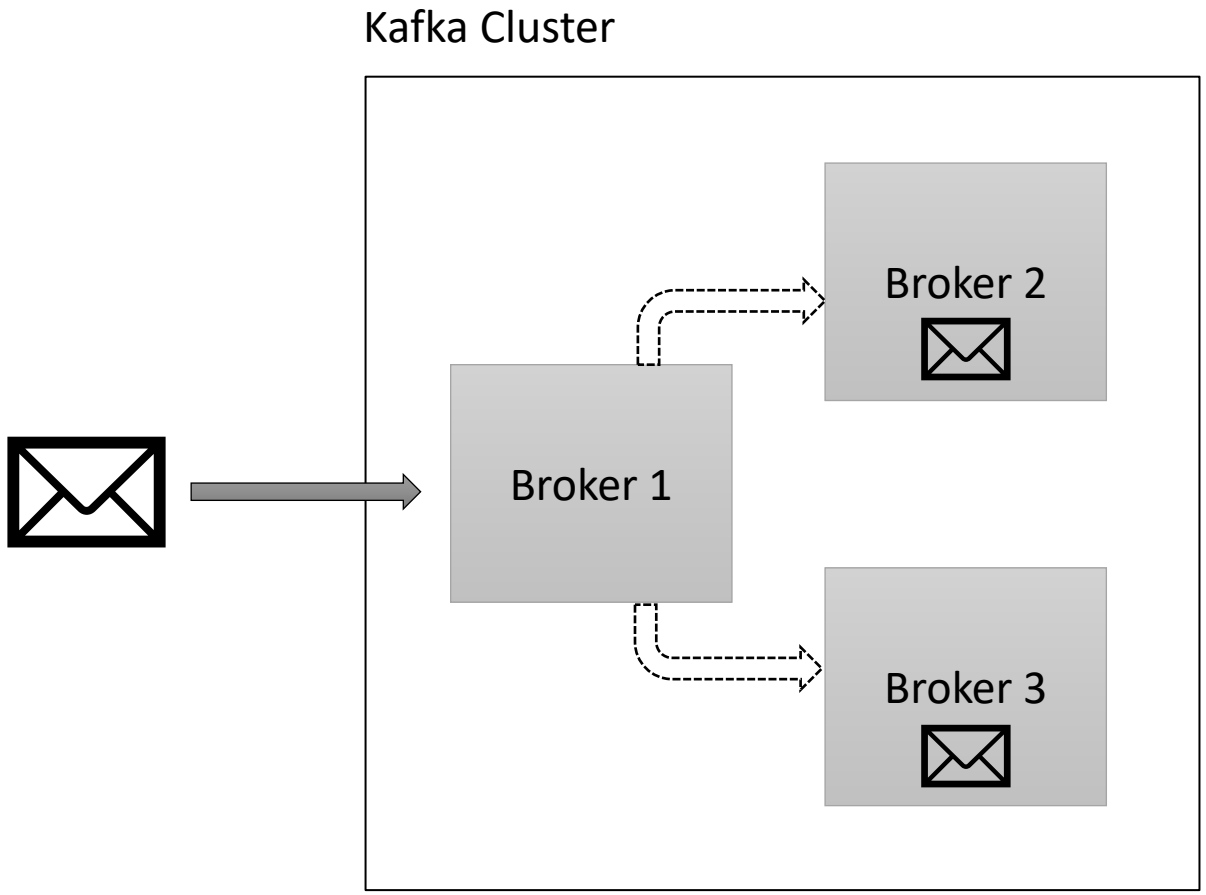
Some of the benefits of using Kafka

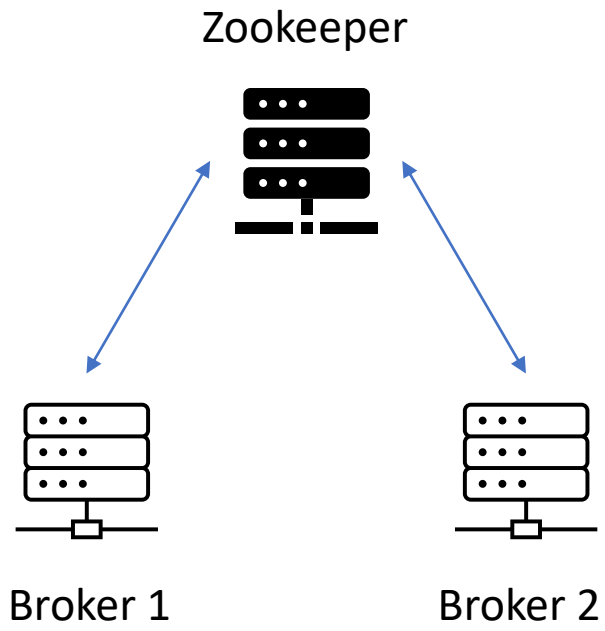
- High throughput
  - No serialization and de-serialization inside Kafka
  - Data durably stored in binary format on the hard disk
- Security
  - Supports data encryption using SSL certificates and TLS
- Scalability
  - Based on a brokered messaging system concept
  - Kafka clusters, Brokers and Zookeeper
  - Publisher subscriber pattern

In contrast to MSMQ

- No distributed transaction support, but established EDA design patterns can be leveraged
- Open source and platform independent
- High availability and easy to scale out both at broker and consumer levels
  - Brokered system as opposed to a federated messaging system

Architecture





Message

- A record stored within Kafka
- Has a Key, Value and a Timestamp
  - **Key**: Can be anything
  - **Value**: Can be anything
  - **Timestamp**: Timestamp when the message was produced. If left out, the producer will automatically add the timestamp
- Key is important when Kafka configuration states the topics to be compacted
- Size limit recommends it to be max 2 MB

Topic

- Ordered collection of messages with a view to categorize the messages
- Stored in the broker
- List of all topics is also managed by the zookeeper
- 2 types of topics
  - **Delete**: Deletes the message based on some configuration like topic size or retention policy of messages
  - **Compacted:** To materialize the view based on the Key. Messages with same key are up-serted, and messages with unique keys inserted

Producer

- Entity that writes messages to a topic in a Kafka cluster
- Deals with concerns like serialization of messages, encrypted communication, network communication etc.
- Writes to a single topic
- To achieve high throughput, have a single producer per application domain, as its expensive to create a producer instance
- When creating you need to define the Key and Value serializer
- When a message is published successfully to the topic, it is said to be “committed to the log”. After that the message will never be lost

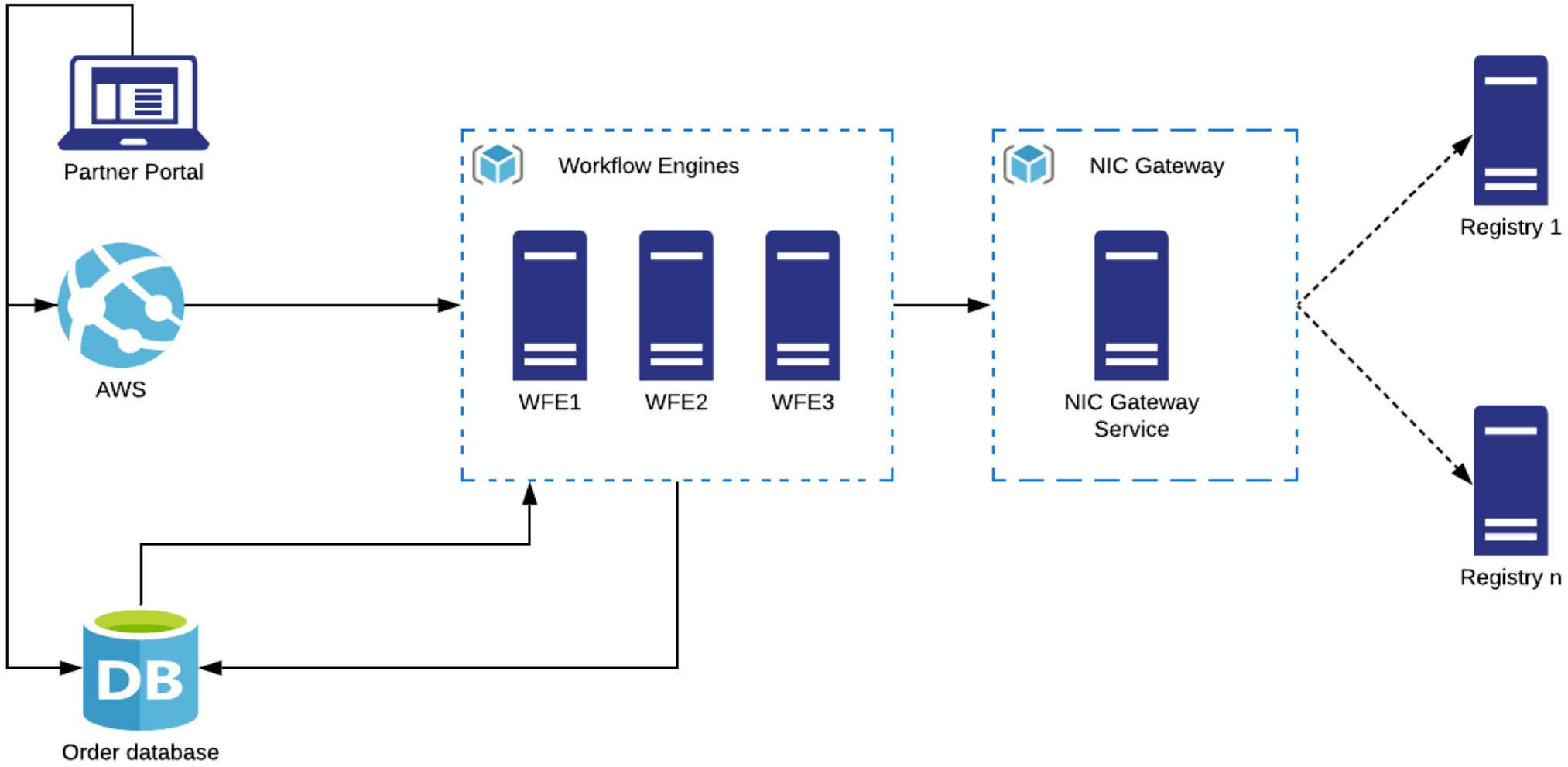
Consumer

- Entity that reads data from one or more Kafka topics
- When creating we need to define the Key and Value de-serializer, which must be the same as the message format
- Works on a PULL mechanism instead of a PUSH mechanism
- The frequency at which to pull messages from topics can be configured (default configuration depends on Kafka library used)
- By default Kafka commits the message offset after reading it, so that its not read again
- For long polling application, this process can be controlled by manually committing the offset after successfully processing the message

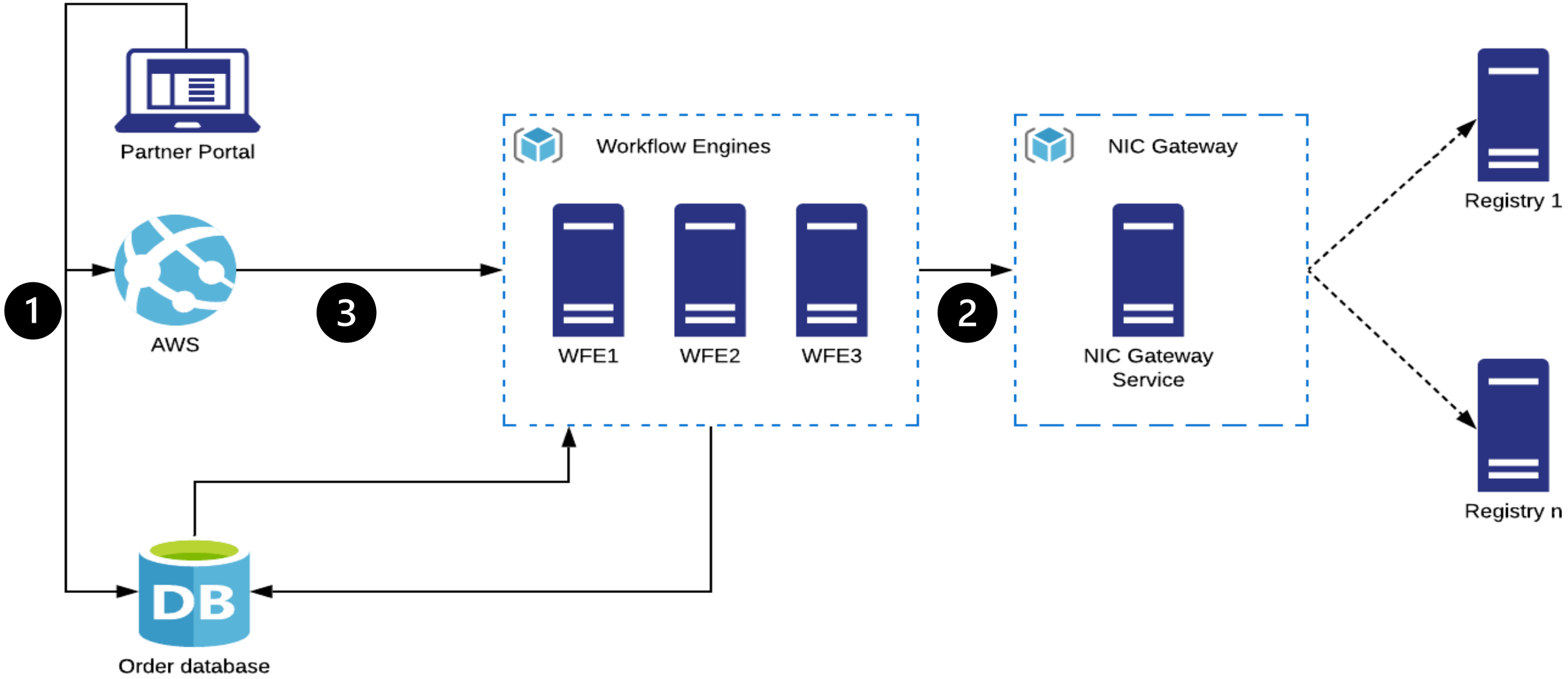
Delivery semantics

- At most once
  - Messages are never re-delivered
  - May not be delivered at all
  - Consumer read -> Commit Offset -> Process
- At least once
  - Never lost
  - May be re-delivered
  - Consumer read -> Process -> Commit offset
- Exactly once
  - Never lost
  - Never re-delivered
  - Transactional producers, Kafka streams, Custom managing consumer offset strategy when using external systems

Ascio Order Processing



Identifying bottlenecks



Discussion on mitigating from the bottlenecks

- How can we eliminate the bottlenecks?
- Building to scale
- Costs
  - Infrastructure
  - Op-Ex
  - Technical skills
  - Maintenance and Support
- More ?

Demo

https://github.com/jbhardwaj-tc/kafka-demo