Message Oriented Middleware

17 November
Lecture 4

Slide Credits: Maarten van Steen

Topics for Today

- Message Oriented Middleware
  - IBM MQ
  - Other systems

- Source: VT 4.2-4.3

Message-Oriented Middleware

Essence: Asynchronous persistent communication through support of middleware-level **queues**. Queues correspond to buffers at communication servers.

Primitive	Meaning
Put	Append a message to a specified queue
Get	Block until the specified queue is nonempty, and remove the first message
Poll	Check a specified queue for messages, and remove the first. Never block
Notify	Install a handler to be called when a message is put into the specified queue

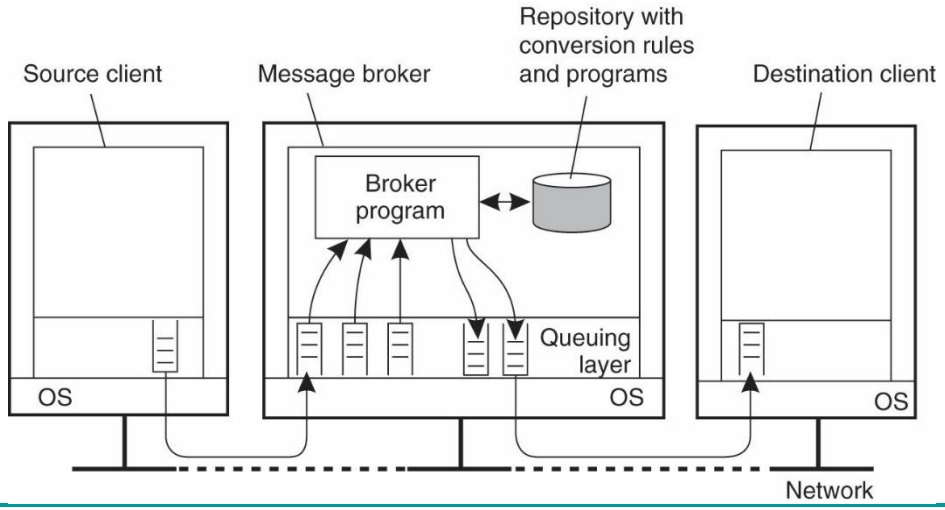
See also: Java Queue Interface (JDK 18)

Message Broker

Observation: Message queuing systems assume a common messaging protocol: all applications agree on message format (i.e., structure and data representation)

Message broker (Integration Bus): Centralized component that takes care of application heterogeneity in an MQ system:

- Transforms incoming messages to target format
- Very often acts as an **application gateway**
- May provide **subject-based** routing capabilities ⇒ **Enterprise Application Integration**



IBM MQ (1/3) (https://www.youtube.com/watch?v=ynjc5GMQeRA)

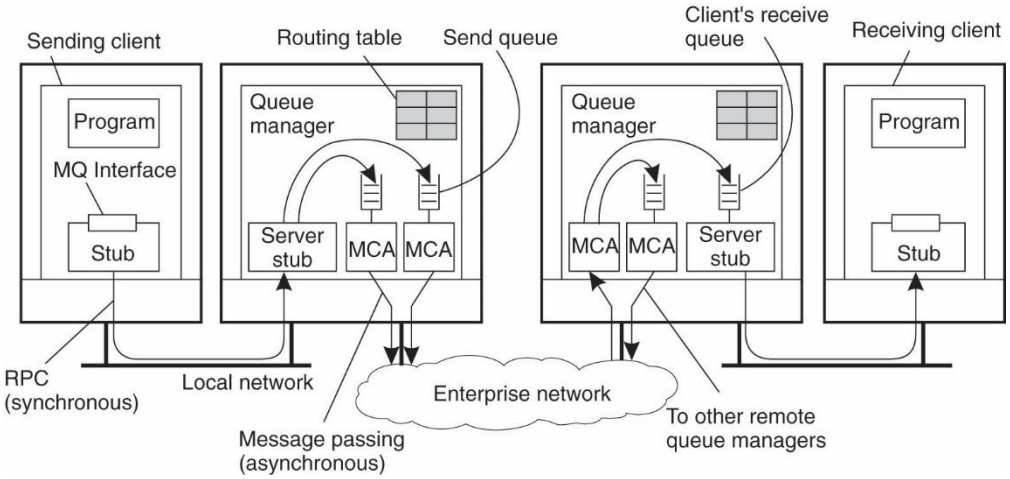
Basic concepts:

- **Application-specific messages** are put into, and removed from **queues**
- Queues always reside under the regime of a **queue manager**
- Processes can put messages only in local queues, or through an RPC mechanism

Message transfer:

- Messages are transferred between queues
- Message transfer between queues at different processes, requires a **channel**
- At each endpoint of channel is a **message channel agent**
- Message channel agents are responsible for:
  - Setting up channels using lower-level network communication facilities (e.g., TCP/IP)
  - (Un)wrapping messages from/in transport-level packets
  - Sending/receiving packets

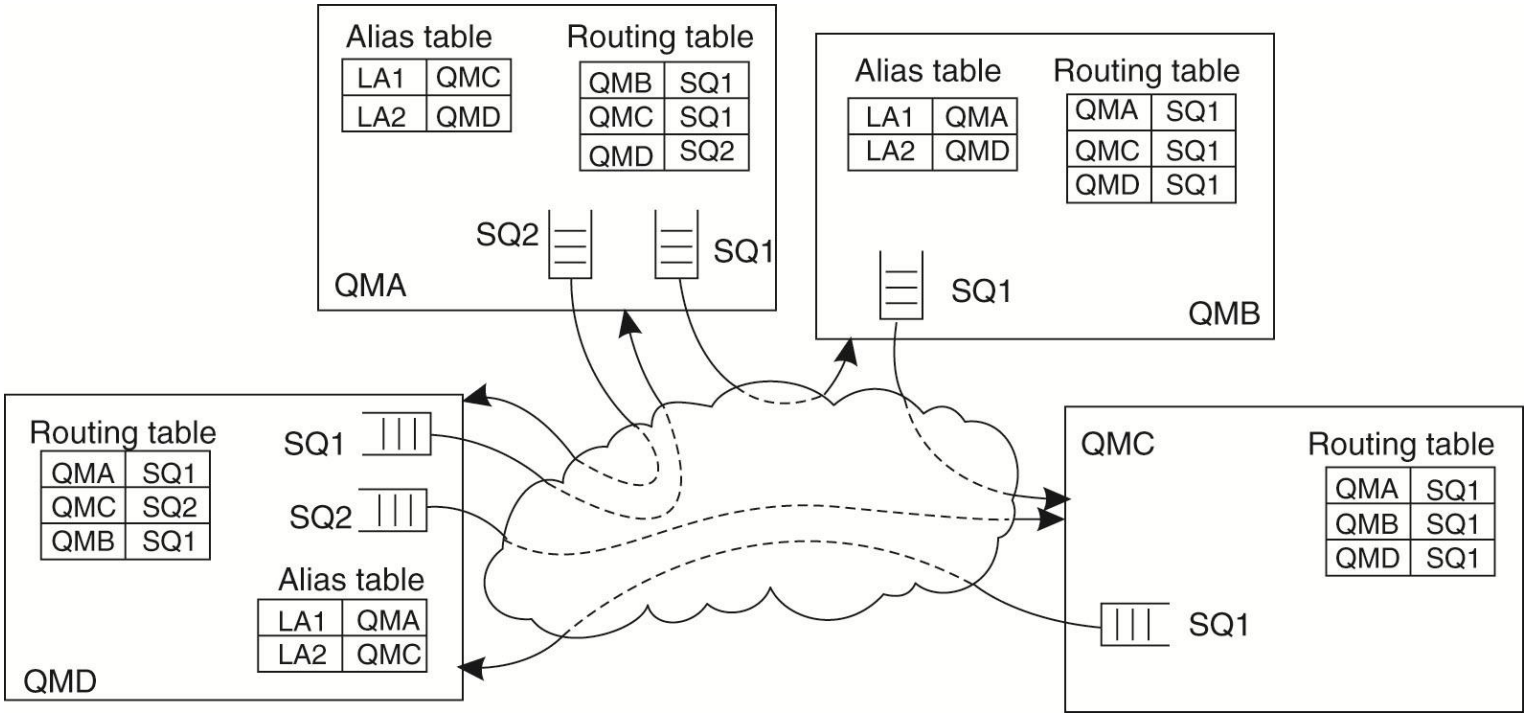
IBM MQ (2/3)



- Channels are inherently unidirectional
- MQ provides mechanisms to automatically start MCAs when messages arrive, or to have a receiver set up a channel
- Any network of queue managers can be created; routes are set up manually (system administration)

IBM MQ (3/3)

Routing: By using **logical names**, in combination with name resolution to local queues, it is possible to put a message in a **remote queue**

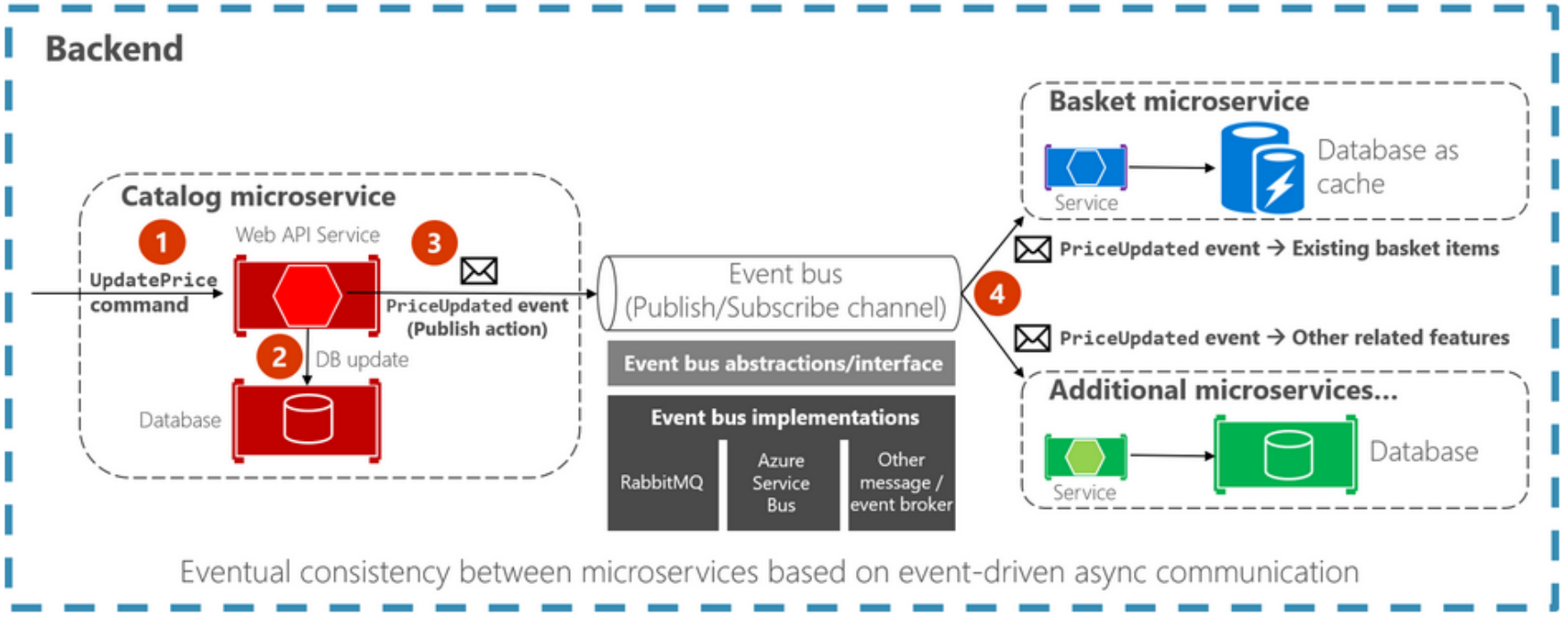


Question: What's a major problem here?

Others: Microsoft

https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/integration-event-based-microservice-communications
https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/subscribe-events

Implementing asynchronous event-driven communication with an event bus



Others: Azure

https://azure.microsoft.com/en-us/services/event-hubs/

Microsoft Azure

Contact Sales: 0201605845 Search My account Portal Si

Overview Solutions Products Documentation Pricing Training Marketplace Partners Support Blog More Free acco

Explore Event Hubs: Pricing details Documentation

Event Hubs is a fully managed, real-time data ingestion service that's simple, trusted, and scalable. Stream millions of events per second from any source to build dynamic data pipelines and immediately respond to business challenges. Keep processing data during emergencies using the geo-disaster recovery and geo-replication features.

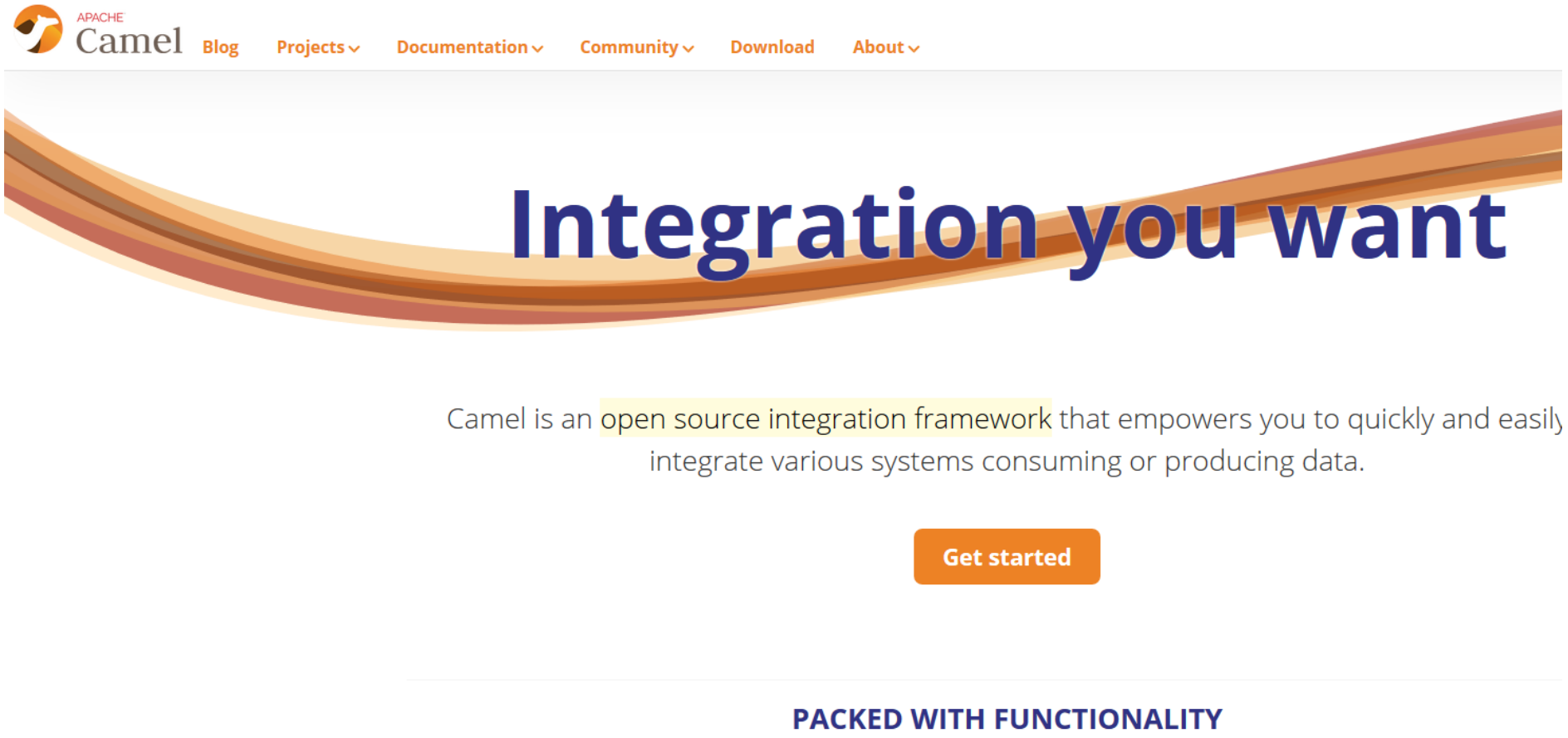
Integrate seamlessly with other Azure services to unlock valuable insights. Allow existing Apache Kafka clients and applications to talk to Event Hubs without any code changes—you get a managed Kafka experience without having to manage your own clusters. Experience real-time data ingestion and microbatching on the same stream.

Link to video >



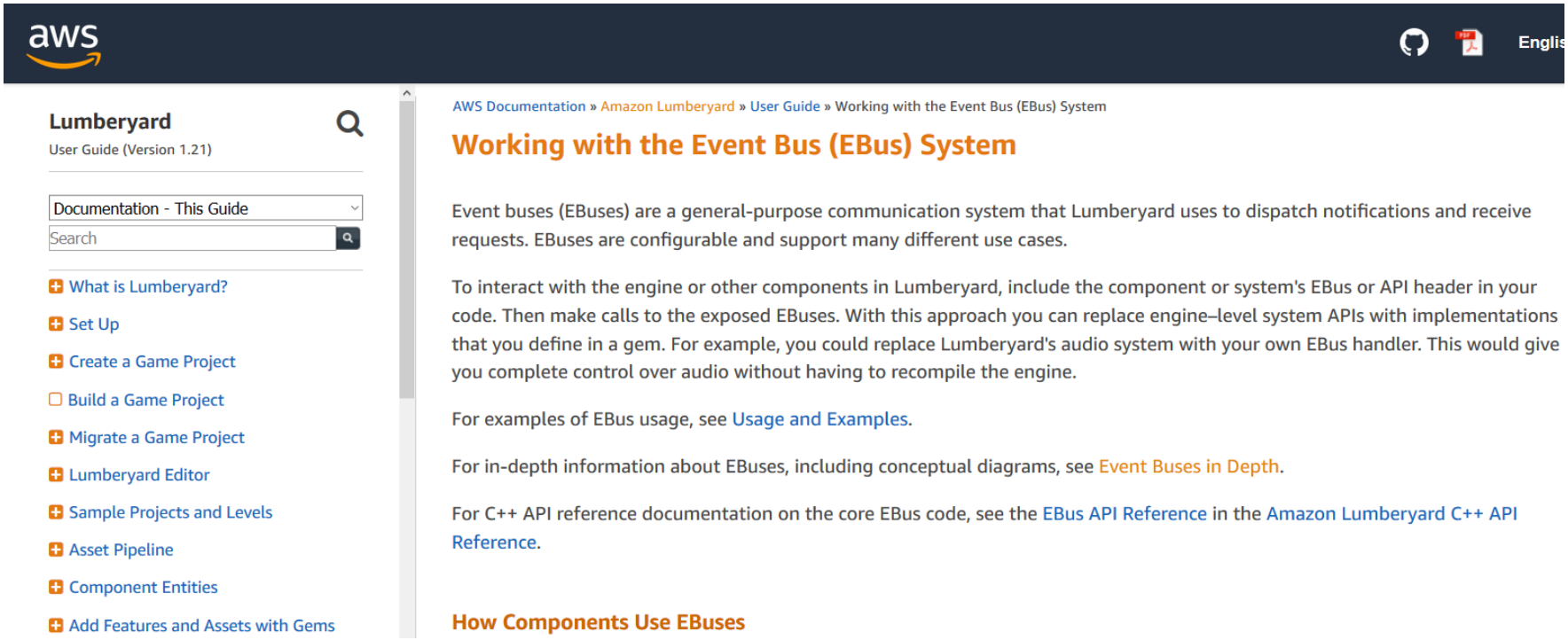
Other: Apache Camel

https://camel.apache.org/



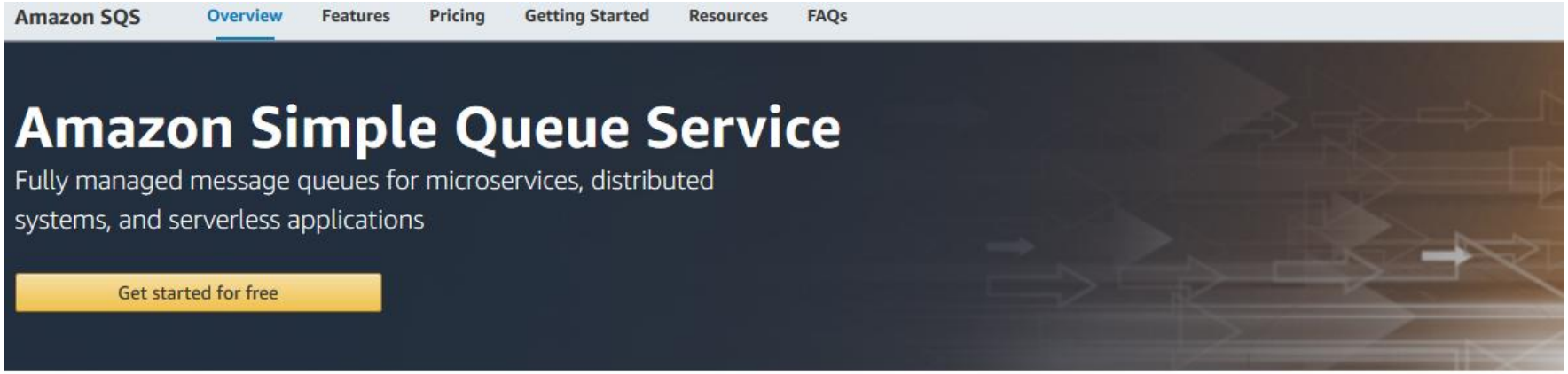
Others: Amazon EBus (Event Bus)

https://docs.aws.amazon.com/lumberyard/latest/userguide/ebus-intro.html



Others: Amazon SQS

https://aws.amazon.com/sqs/



Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware, and empowers developers to focus on differentiating work. Using SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available. Get started with SQS in minutes using the AWS console, Command Line Interface or SDK of your choice, and three simple commands.

SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.



Conclusion

- Message Oriented Middleware
  - IBM MQ
  - Other systems