

AZURE SERVICE BUS



MVP CONF
2025

Felipe Pimentel Augusto

- Formado em Ciências da Computação @ UGF
- Pós Graduado em MIT Arquitetura de Sistemas @ INFNET
- Aluno externo de Mestrado @ UTFPR
- Eterno Estudante e entusiasta de tecnologias
- +20 anos de carreira ligadas à desenvolvimento, arquitetura e liderança técnica
- Technical Architecture Manager @ Avanade
- Especialista em .NET, Azure, Containers e EDA (Event-Driven Architecture)
- Possuo algumas certificações em Azure, AWS, GitHub e outros vendors
- Apaixonado por sistemas críticos e de alta performance
- Experiência em indústrias de Health, Financial, Resources e Education
- Microsoft MVP em Developer Technology, .NET
- Host no canal DEPLOY @ Youtube





>Agenda

- Comando/Command
- Evento/Event
- Event Storm or Evento Modeling
- Event Driven
- Event Sourcing
- Comparativos
- Azure Service Bus
 - Filas/Queue
 - Tópicos/Topic
 - Partição
 - Sessão
 - DeadLetterQueue
 - Cloud Events



Comando

Definição: Um **comando** representa uma **intenção de ação** no sistema.

📌 **Característica principal:** Pode ser **aceito ou rejeitado**.

❖ **Exemplo:**

- "Criar um pedido"
- "Atualizar o endereço do usuário"
- "Cancelar uma assinatura"

📌 **Pontos-chave sobre comandos:**

✓ **Imperativo:** Ele manda o sistema executar uma ação.

✓ **Nomeado como verbo no infinitivo:** Exemplo:

"CriarPedido", "ProcessarPagamento".

✓ **Pode ser rejeitado:** Se o comando não for válido (ex: saldo insuficiente), ele pode falhar.

✓ **Destinatário único:** Um comando geralmente é enviado para um **único serviço** responsável.

```
1  ✓ {  
2   "type": "CriarPedido",  
3   "clientId": 123,  
4   "itens": [  
5     {  
6       "SKU": "AC4643",  
7       "quantidade": 1  
8     }  
9   ]  
10 }
```



Evento

📌 **Definição:** Um **evento** representa **algo que já aconteceu** no sistema.

📌 **Característica principal:** Ele **não pode ser revertido**.

◆ **Exemplo:**

- "Pedido Criado"
- "Pagamento Confirmado"
- "Email Enviado"

•📌 **Pontos-chave sobre eventos:**

✓ **Descritivo:** Ele informa que algo aconteceu.

✓ **Não pode ser rejeitado:** O evento já ocorreu, então ele é sempre válido.

✓ **Nomeado no passado:** Exemplo: "PedidoCriado", "PagamentoAprovado".

✓ **Pode ter múltiplos consumidores:** Vários serviços podem reagir a um evento.

```
1 {  
2   "type": "PedidoCriado",  
3   "pedidoId": "456",  
4   "clienteId": 123,  
5   "data": "2025-03-27T10:00:00Z"  
6 }
```



Comando e Evento

Característica	Comando (Command)	Evento (Event)
O que é?	Uma instrução para fazer algo	Um fato que já aconteceu
Pode ser rejeitado?	Sim	Não, pois já aconteceu
Forma verbal	Verbo no infinitive “CriarPedido”	Verbo no passado “Pedido Criado”
Sincronia	Pode ser síncrono ou assíncrono	Sempre Assíncrono
Consumidores	Apenas um destinatário	1:n assinantes (subscribers)



Comando e Evento + Event-Driven e Event Sourcing?

Comandos são usados para **solicitar** mudanças no sistema.

Eventos são usados para **notificar** que mudanças já ocorreram.

Arquitetura Event-Driven depende de eventos para permitir comunicação assíncrona entre serviços.

Event Sourcing armazena eventos como fonte primária de verdade do sistema.



Event Storming ou Event Modeling

- ◆ O que é?

Event Storming é uma **técnica de modelagem colaborativa** usada para entender e mapear os eventos de um domínio de negócio.

- ◆ Para que serve?

- Descobrir fluxos de eventos dentro de um sistema
- Identificar **comandos, agregados, leitores e escritores**
- Facilitar a comunicação entre times técnicos e de negócio

- ◆ Como funciona?

- Utiliza **post-its coloridos** para representar eventos, comandos, atores e interações
- Geralmente, é feito **em grupo** com desenvolvedores, POs, designers e stakeholders
- Ajuda a visualizar gargalos, otimizar processos e criar uma **visão clara do domínio**



Event Driven

- ◆ O que é?

Event-Driven é um **padrão arquitetural** onde os sistemas reagem a eventos ao invés de funcionarem com chamadas diretas síncronas.

- ◆ Para que serve?

- Criar sistemas assíncronos e desacoplados
- Melhorar escalabilidade e resiliência
- Usado em **mensageria** (ex: Azure Service Bus, Kafka, RabbitMQ)
- Viabiliza **Event Sourcing** e **CQRS**

- ◆ Como funciona?

- Os serviços produzem eventos e os publicam em um barramento de mensagens
- Outros serviços podem consumir esses eventos e reagir a eles
- O fluxo é assíncrono e baseado em pub/sub (publish-subscribe)



Event Sourcing

- ◆ **O que é?**

Event Sourcing é um **padrão arquitetural** onde o estado de um sistema não é armazenado diretamente em bancos de dados tradicionais, mas sim reconstruído a partir de uma **sequência de eventos passados**.

Em vez de armazenar apenas o estado atual de uma entidade (como uma tabela no banco de dados com a última versão do pedido), **todos os eventos que levaram a esse estado são armazenados e podem ser "replayados" para reconstruí-lo.**



Event Sourcing

Como Funciona?

1 Captura de Eventos

Toda vez que uma ação relevante ocorre no sistema, um **evento imutável** é registrado.

Exemplo: "Pedido Criado", "Pagamento Aprovado", "Pedido Enviado".

2 Armazenamento dos Eventos

Os eventos são armazenados em um **Event Store** (banco especializado ou log de eventos).

Diferente de um banco relacional tradicional, cada evento é **imutável** e ordenado cronologicamente.

3 Reconstrução do Estado

Em vez de armazenar diretamente o estado atual de uma entidade, o sistema pode "**replayar**" os eventos para recalcular o estado.

Isso permite auditar mudanças e reconstruir estados passados.

4 Projeções e Query Models (CQRS)

Muitas vezes, é usado junto com **CQRS (Command Query Responsibility Segregation)**, onde **as consultas são otimizadas através de projeções** que transformam eventos em modelos prontos para leitura.



> Outros serviços de mensageria



- Azure Service Bus



- Event Grid



- Event Hub



- Storage Account



Azure Service Bus

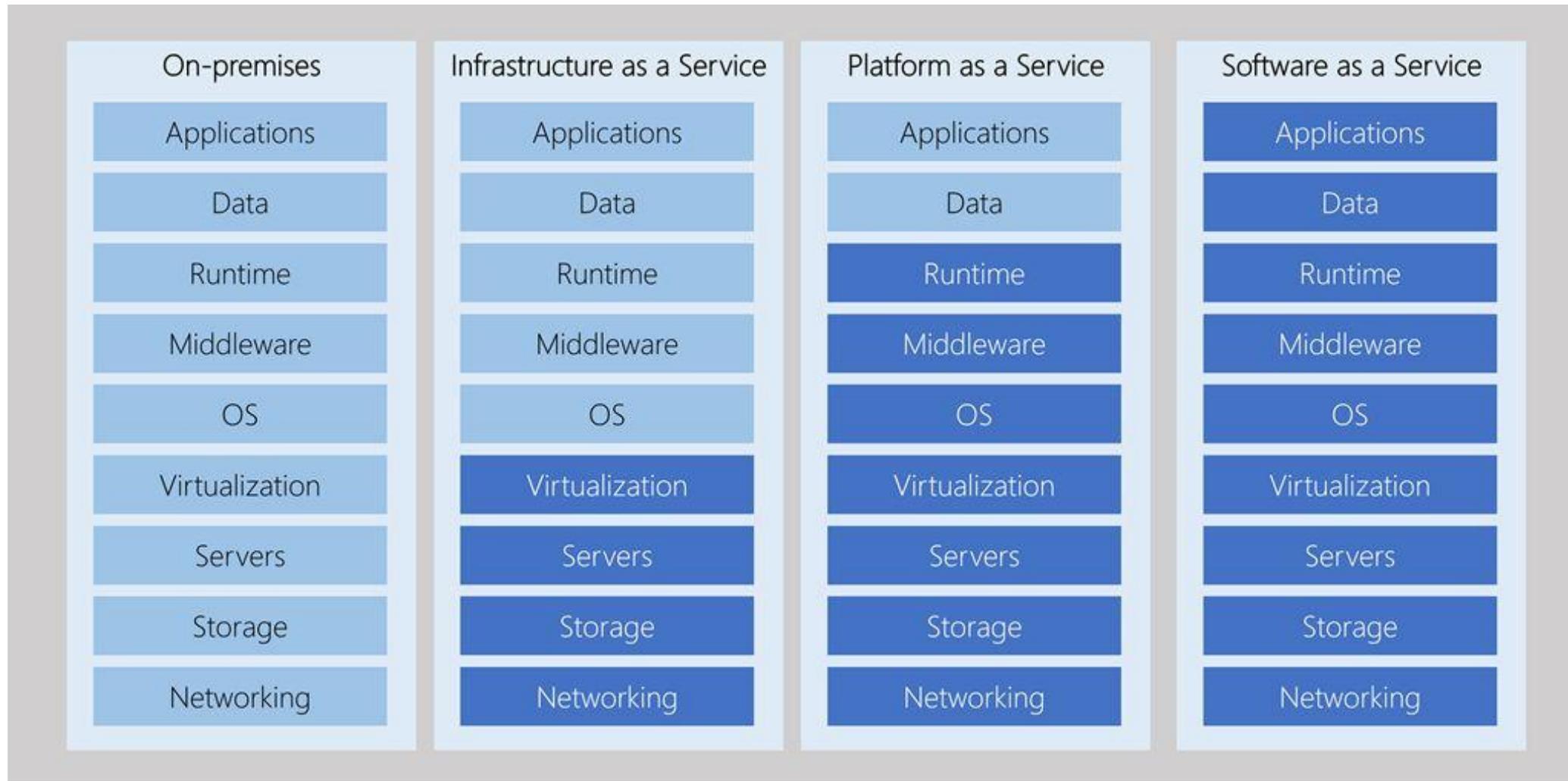
É um agente de mensagens totalmente gerenciado com filas de mensagens e tópicos de publicação/assinatura.

É usado para separar aplicativos e serviços uns dos outros, proporcionando estes benefícios:

- Balanceamento da carga de trabalho entre funções de trabalho concorrentes
- Roteamento e transferência de dados e do controle, com segurança, entre limites de serviços e aplicativos
- Coordenação do trabalho transacional que requer um alto grau de confiabilidade



Azure Service Bus



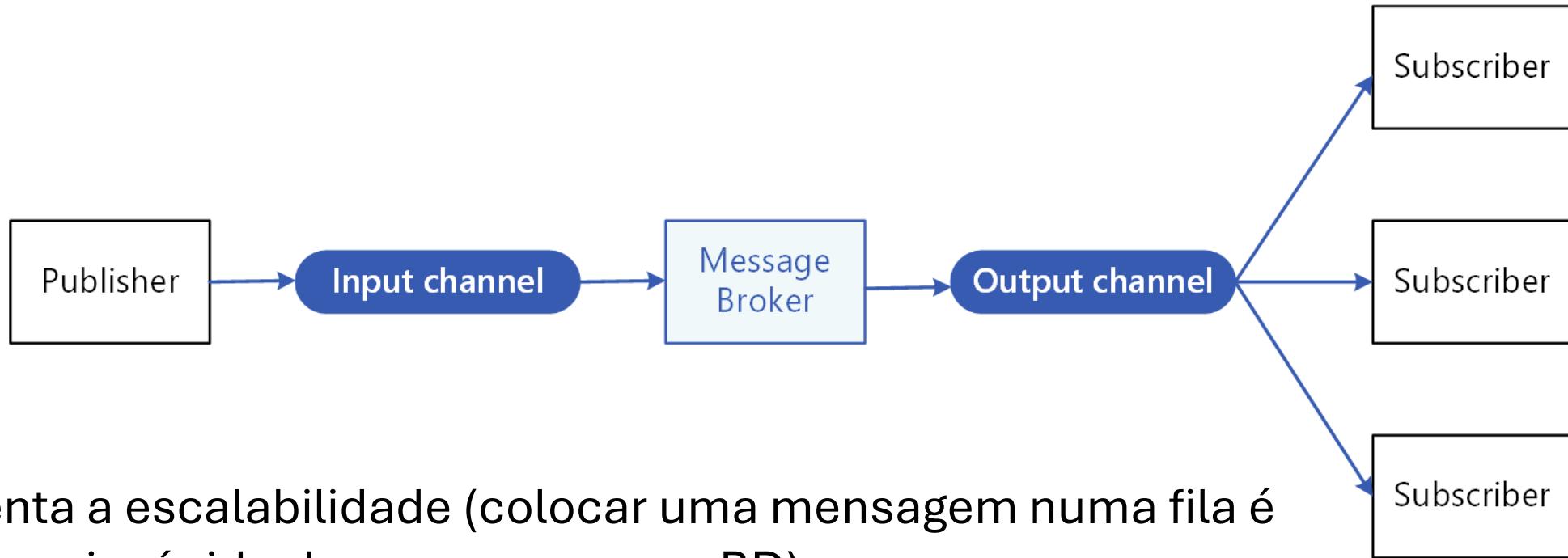
Fila / Queue



Tópico / Topic



Pub/Sub Pattern



Aumenta a escalabilidade (colocar uma mensagem numa fila é muito mais rápido do que escrever no BD);

Aumenta confiabilidade (mensagens assíncronas aguentam o aumento da carga de trabalho);

Permite processamento agendado ou adiado;

Permite integração entre sistemas com plataformas diferentes;

Melhora capacidade de testes;

Facilita a separação de responsabilidades



Assinante/Consumer/Subscriber



Partição

Divisão de tópicos ou filas em partes menores para melhorar a escalabilidade e o desempenho.

- O que são : As mensagens são distribuídas entre partições, permitindo paralelismo e balanceamento de carga.
- Como funciona : Cada partição é independente e pode ser processada por consumidores diferentes.
- Vantagens : Alta disponibilidade, maior taxa de transferência e tolerância a falhas.
- Uso típico : Aplicações com alto volume de mensagens que exigem distribuição eficiente e confiabilidade.



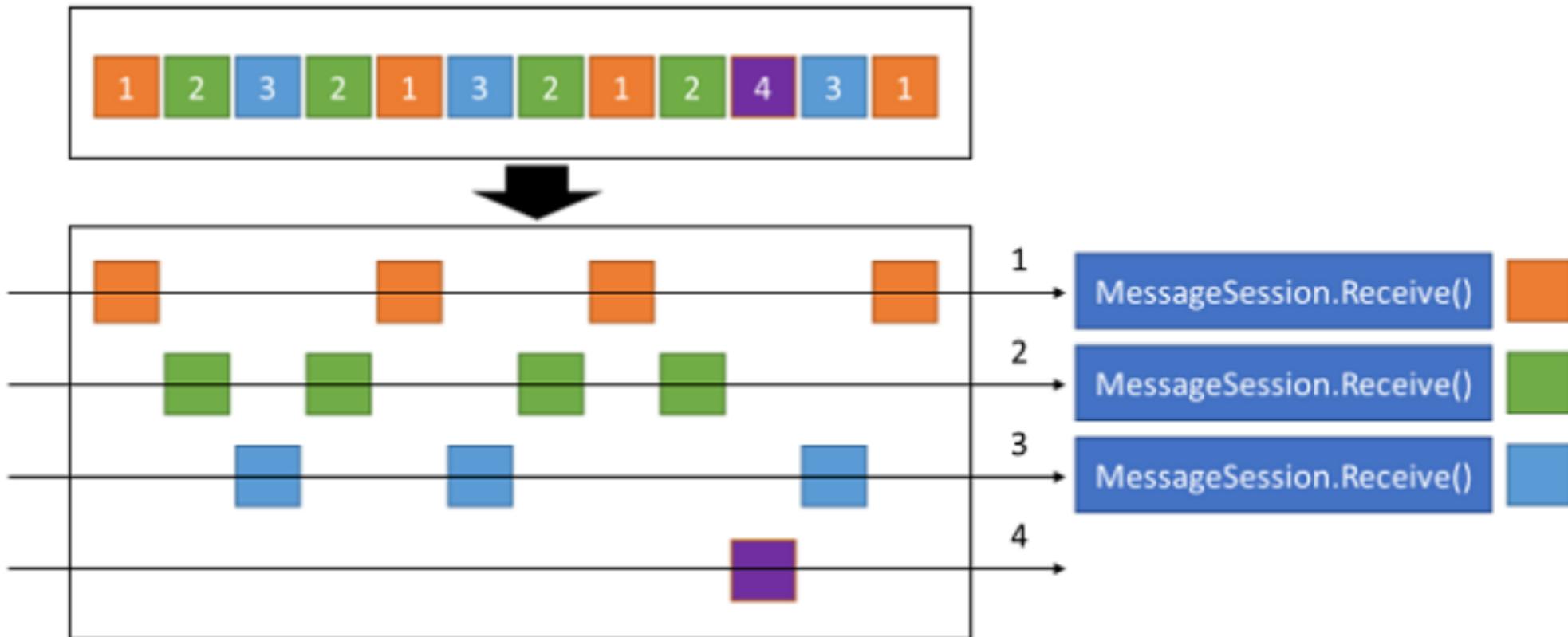
Sessão

Recursos que permitem processar mensagens de forma ordenada e agrupada.

- O que são : Sessões garantem que um grupo de mensagens seja processado por um único consumidor, mantendo a ordem.
- Como funciona : Mensagens com o mesmo *Session ID* são tratadas como parte da mesma sessão.
- Uso típico : Cenários como carrinhos de compras, pedidos ou qualquer fluxo que exija processamento sequencial de dados relacionados.
- Benefícios : Ordenação garantida, controle de estado e processamento consistente.



Sessão



DeadLetter

Fila especial para mensagens não processáveis após várias tentativas.

- Não podem ser processadas com sessão
- Causas comuns : Erros de processamento, expiração ou filtros inválidos.
- Benefícios : Permite análise, retentativa ou arquivamento seguro.
- Uso típico : Identificar e corrigir falhas no fluxo de mensagens.



Cloud Events

CloudEvents é uma especificação para descrever dados de eventos de forma comum. O CloudEvents busca simplificar drasticamente a declaração e a entrega de eventos em serviços, plataformas e muito mais!

O grupo de trabalho CloudEvents tem despertado grande interesse da indústria, desde grandes provedores de nuvem até empresas de SaaS populares. A especificação agora está sob a tutela da Cloud Native Computing Foundation.

Padrão aberto para descrever eventos de forma consistente em sistemas distribuídos.

- O que é : Um formato padronizado (baseado em JSON) para estruturar dados de eventos, facilitando a interoperabilidade entre serviços e plataformas.
- Vantagens : Simplifica a integração entre sistemas diferentes, promove portabilidade e reduz ambiguidades.
- Uso típico : Eventos em microserviços, streaming de dados e integração entre nuvens ou plataformas.



Queue <> Topic

Quesitos	Queue	Topic
Conceito	Mensagens são entregues a um único consumidor (First In First Out).	Mensagens são entregues a vários consumidores via assinaturas.
Modelo de Comunicação	Um para um: Uma mensagem é processada por apenas um receptor.	Um para muitos: Uma mensagem pode ser processada por múltiplos receptores.
Escalabilidade	Ideal para cenários simples de fila de trabalho.	Ideal para cenários de publicação/assinatura (pub/sub).
Assinaturas	Não possui assinaturas.	Suporta várias assinaturas cada uma com sua própria lógica de filtro.
Filtros	Não há suporte nativo para filtragem.	Suporta filtros baseados em propriedades da mensagem.
Uso Típico	Processamento sequencial de tarefas (ex.: processamento de pedidos).	Distribuição de eventos para múltiplos sistemas ou serviços (ex.: notificações).

Message Construct.

Message
Command Message
Document Message
Event Message
Request-Reply
Return Address
Correlation Identifier
Message Sequence
Message Expiration
Format Indicator

Message Routing

Pipes-and-Filters
Message Router
Content-based Router
Message Filter
Dynamic Router
Recipient List
Splitter
Aggregator
Resequencer
Composed Msg. Processor
Scatter-Gather
Routing Slip
Process Manager
Message Broker

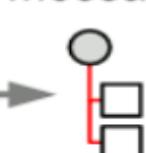
Message Transformation

Message Translator
Envelope Wrapper
Content Enricher
Content Filter
Claim Check
Normalizer
Canonical Data Model

Endpoint

Message

Application A



Channel

Router

Translator

Endpoint

Application B

Messaging Endpoints

Message Endpoint
Messaging Gateway
Messaging Mapper
Transactional Client
Polling Consumer
Event-driven Consumer
Competing Consumers
Message Dispatcher
Selective Consumer
Durable Subscriber
Idempotent Receiver
Service Activator

Messaging Channels

Message Channel
Point-to-Point Channel
Publish-Subscr. Channel
Datatype Channel
Invalid Message Channel
Dead Letter Channel
Guaranteed Delivery
Channel Adapter
Messaging Bridge
Message Bus

Systems Mgmt.

Control Bus
Detour
Wire Tap
Message History
Message Store
Smart Proxy
Test Message
Channel Purger



It's Tiiiiimmeeeeeee



Referencias

- Github
 - <https://github.com/felipementel/DEPLOY.AzureServiceBus>
- Boas Práticas para nomenclaturas e abreviações no Azure
 - <https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming>
 - <https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/ready/azure-best-practices/resource-abbreviations>
- Design Patterns
 - <https://learn.microsoft.com/pt-br/azure/architecture/patterns/publisher-subscriber>
 - <https://learn.microsoft.com/pt-br/azure/architecture/patterns/competing-consumers>
 - <https://learn.microsoft.com/pt-br/azure/architecture/patterns/event-sourcing>
 - <https://learn.microsoft.com/pt-br/azure/architecture/patterns/cqrs>
 - <https://microservices.io/patterns/data/transactional-outbox.html>
 - <https://learn.microsoft.com/pt-br/azure/architecture/patterns/saga>
 - <https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-resource-manager-exceptions>
 - <https://cloudevents.io>
 - <https://learn.microsoft.com/en-us/samples/azure-samples/cosmos-db-design-patterns/event-sourcing/>
- Componentes:
 - <https://www.enterpriseintegrationpatterns.com/patterns/messaging/index.html>
- Exemplos de código
 - <https://learn.microsoft.com/pt-br/samples/azure/azure-sdk-for-net/azurermessagingservicebus-samples/>

