

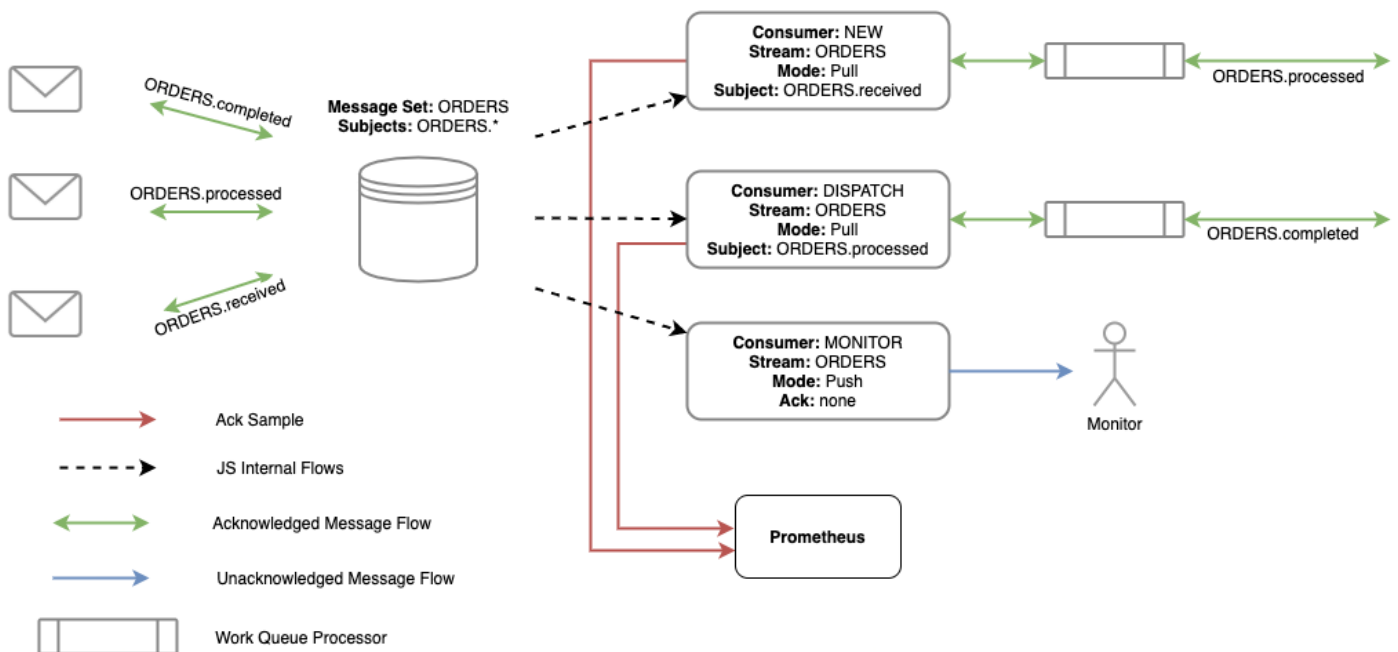
Example



- [Example Configuration](#)

Consider this architecture

JetStream mixed push/pull Order processing architecture



While it is an incomplete architecture it does show a number of key points:

- Many related subjects are stored in a Stream
- Consumers can have different modes of operation and receive just subsets of the messages
- Multiple Acknowledgement modes are supported

A new order arrives on **ORDERS.received**, gets sent to the **NEW** Consumer who, on success, will create a new message on **ORDERS.processed**. The **ORDERS.processed** message again enters the Stream where a **DISPATCH** Consumer receives it and once processed it will create an **ORDERS.completed** message which will again enter the Stream. These operations are all **pull** based meaning they are work queues and can scale horizontally. All require acknowledged delivery ensuring no order is missed.

All messages are delivered to a `MONITOR` Consumer without any acknowledgement and using Pub/Sub semantics - they are pushed to the monitor.

As messages are acknowledged to the `NEW` and `DISPATCH` Consumers, a percentage of them are Sampled and messages indicating redelivery counts, ack delays and more, are delivered to the monitoring system.

Example Configuration

[Additional documentation](#) introduces the `nats` utility and how you can use it to create, monitor, and manage streams and consumers, but for completeness and reference this is how you'd create the ORDERS scenario. We'll configure a 1 year retention for order related messages:

```
nats stream add ORDERS --subjects "ORDERS.*" --ack --max-msgs=-1 --max-bytes=-1 --max-age=1y
--storage file --retention limits --max-msg-size=-1 --discard=old
nats consumer add ORDERS NEW --filter ORDERS.received --ack explicit --pull --deliver all --
max-deliver=-1 --sample 100
nats consumer add ORDERS DISPATCH --filter ORDERS.processed --ack explicit --pull --deliver
all --max-deliver=-1 --sample 100
nats consumer add ORDERS MONITOR --filter '' --ack none --target monitor.ORDERS --deliver last
--replay instant
```

📄 #3

📄 cocobook-sync 📄 2026-10-28 11:10:26 UTC

📄 cocobook-sync 📄 2026-10-29 03:58:09 UTC