Pub-Sub Project

Learning Objectives

Build a production-ready pub-sub system to learn:

- Fan-out pattern (one message → many subscribers)
- Topic-based routing
- Slow consumer handling (backpressure)
- At-least-once delivery guarantees
- Graceful shutdown with message draining

Requirements

Functional

1. **Publishing**
   - Publish to topic
   - Message buffering
   - Async delivery (don't block publisher)
2. **Subscribing**
   - Subscribe to topic patterns
   - Multiple subscribers per topic
   - Ordered delivery per subscriber
3. **Reliability**
   - Persistent queue (survive restarts)
   - At-least-once delivery
   - Message acknowledgment

Non-Functional

1. **Performance** - >10k msgs/sec
2. **Scalability** - 1000+ concurrent subscribers
3. **Reliability** - No message loss on crash

Implementations

Naive

In-memory only, blocking publish, no slow consumer handling

Improved

Buffered channels per subscriber, non-blocking publish, cleanup

Final

Persistent storage, at-least-once, circuit breaker, metrics

Usage

```go
ps := final.NewPubSub(final.Config{
    PersistPath: "./messages.db",
    BufferSize: 1000,
})

// Publish
ps.Publish("orders", Order{ID: 123})

// Subscribe
sub := ps.Subscribe("orders")
for msg := range sub {
    processOrder(msg)
    msg.Ack() // Acknowledge receipt
}
```