1. Design Document

Why this approach?

The **Quorum-Based Lease** approach was chosen because it provides the strongest safety guarantees in a "Chaos" environment. By eliminating a leader node, we eliminate the "Single Point of Failure." Every node operates independently with its own SQLite state, making the system naturally resistant to individual node crashes.

Handling Split-Brain

In a 3-node cluster, a "Split-Brain" scenario (where the network is cut in half) is handled by the Quorum requirement.

- **The Minority Side:** A client connected to only 1 node will fail to reach the threshold of 2 votes and will be denied the lock.
- **The Majority Side:** Clients connected to the remaining 2 nodes will successfully acquire the lock.
  This ensures that even during total network partitioning, only one "brain" can be active at a time.

Handling Clock Skew

We address the unreliability of distributed clocks using a two-pronged strategy:

1. **Monotonicity:** On the server side, we ignore the system wall clock. By using a monotonic counter, we ensure that a 10-second lease is exactly 10 seconds of hardware time, even if the system's "Time of Day" jumps backward or forward.
2. **Client-Side Drift Compensation:** The client implements "Pessimistic Validity." It assumes that the servers may have a slight drift. By artificially shortening its own lease window and accounting for network trip time, the client guarantees it will stop using a resource *before* any server considers the lock available for the next requester.

What could still go wrong?

- **Storage Exhaustion:** If the disk on all nodes becomes full, the SQLite persistence layer will fail to record grants.
- **Extreme Execution Pauses:** If a client process experiences a Garbage Collection (GC) or OS pause that exceeds the TTL and the Drift Buffer combined, the client may believe it still holds the lock when it wakes up. To mitigate this, the application must use the provided **Fencing Token** to validate writes at the storage layer.