**ch 1**

**Reliable, Scalable, and Maintainable Applications**

**Theme:** Reliable, Scalable, and Maintainable Applications

**1. Core Ideas**

* Modern apps are **data-intensive** (bottleneck = data volume, speed, complexity; not raw CPU).
* Built from **standard components**:
  + Databases (persistent storage)
  + Caches (speed reads)
  + Search indexes (keyword filtering)
  + Stream processors (async messaging)
  + Batch processors (periodic crunching)
* Goal: achieve **reliability, scalability, maintainability** — the three pillars of DDIA.

**2. Reliability**

* A reliable system **continues to work correctly even with faults**.
* Threats:
  + **Hardware faults** → disks crash, RAM fails, networks partition.
  + **Software errors** → bugs, memory leaks, deadlocks.
  + **Human errors** → misconfigs, bad deployments.
* Countermeasures:
  + Replication & redundancy.
  + Fault isolation (bulkheads).
  + Automated recovery (self-healing infra).
* ⚖️ Trade-off: Reliability ≠ 100% uptime → cost vs probability of failure.

**Mapping to GenAI Pyramid / LexiFlow:**

* **Infra (Layer 1):** ECS Fargate in multi-AZ + ALB failover = reliability.
* **Pattern:** Circuit breakers + rollback runbooks.
* **LexiFlow Example:** If retriever service crashes → ECS auto-restart + fallback to cached context.

**3. Scalability**

* A system is scalable if it can **handle growth in load** by adding resources.
* Define load with **load parameters** (QPS, dataset size, concurrent users).
* Metrics:
  + **Throughput** (ops/sec)
  + **Latency** (response time distribution, not just averages).
* Approaches:
  + Vertical scaling (bigger machine).
  + Horizontal scaling (more machines).
  + Partitioning/sharding data.
* ⚖️ Trade-off: Simplicity (scale-up) vs flexibility & resilience (scale-out).

**Mapping to GenAI Pyramid / LexiFlow:**

* **Infra:** Autoscaling ECS tasks with SQS buffering.
* **RAG (Layer 2):** Shard FAISS/Weaviate indexes for parallel retrieval.
* **LexiFlow Example:** Handle 10x document ingestion by splitting across SQS → workers.

**4. Maintainability**

* Maintainable = system stays **operable, simple, evolvable**.
* Dimensions:
  + **Operability** → ease of monitoring, incident response (CloudWatch, dashboards).
  + **Simplicity** → reduce accidental complexity (clear APIs, modular design).
  + **Evolvability** → adapt to new requirements without massive rewrites (schema evolution, modular pipelines).
* ⚖️ Trade-off: More abstraction = more flexibility but sometimes less performance.

**Mapping to GenAI Pyramid / LexiFlow:**

* **Eval & Guardrails (Layer 4):** Monitoring, A/B testing, rollback runbooks = evolvability.
* **Pattern:** Shadow deploys, cost/latency dashboards.
* **LexiFlow Example:** CloudWatch + structured logs for RAG queries, versioned embeddings for evolution.

**5. Interview-Style Q&A**

* **Q:** What are the 3 pillars of data systems?  
  **A:** Reliability (handles faults), Scalability (handles growth), Maintainability (easy to evolve/operate).
* **Q:** How would you ensure reliability in a RAG system?  
  **A:** ECS task replication, circuit breakers, SQS buffering, fallback to cached embeddings.
* **Q:** How do you describe load in LexiFlow?  
  **A:** Load params = documents ingested/sec, retrieval QPS, concurrent queries.
* **Q:** What’s the maintainability story for LexiFlow?  
  **A:** Modular DAGs (retrieval, rerank, summary), schema-evolvable embeddings, observability dashboards.

**6. Quick Mapping Table (Chapter 1 → Pyramid → LexiFlow)**

| **DDIA Concept** | **Pyramid Layer** | **LexiFlow Application** |
| --- | --- | --- |
| Reliability | Infra | ECS multi-AZ, ALB failover, retries |
| Scalability | Infra + RAG | Autoscaling ECS + FAISS sharding |
| Maintainability | Eval & Guardrails | CloudWatch, rollback runbooks, eval dashboards |
| Load Metrics | Infra | Define QPS, latency for LexiFlow queries |
| Human Error | Infra | Safe deploys → Blue/Green, IAM least privilege |