

Scaling LinkedIn's Search Infrastructure

Key Decisions and Engineering Challenges

Rahul Raja, Staff Software Engineer



What is LinkedIn's Scale ?



**1.2 Billion
members**

Total LinkedIn members



66M visitors

Number of visitors for LinkedIn
Jobs every week



**3.2M Peak
QPS**

Across all LinkedIn data and
systems

Components of a Search Infrastructure

Indexing Engine

- Inverted Index (maps terms to document IDs)
- Forward Index (stores metadata and fields for fast retrieval)

Ranking and Relevance engine

- TF-IDF / BM25 / Vector Similarity
- Learning to Rank Models (ML-based ranking)
- Business Rules (e.g., boosting premium content)

Live Indexing engine

- Live indexing updates the search index in real time as data changes.
- Enables low-latency ingestion from sources like databases or Kafka.
- Ensures search results stay fresh and up-to-date for users.

Serving engine

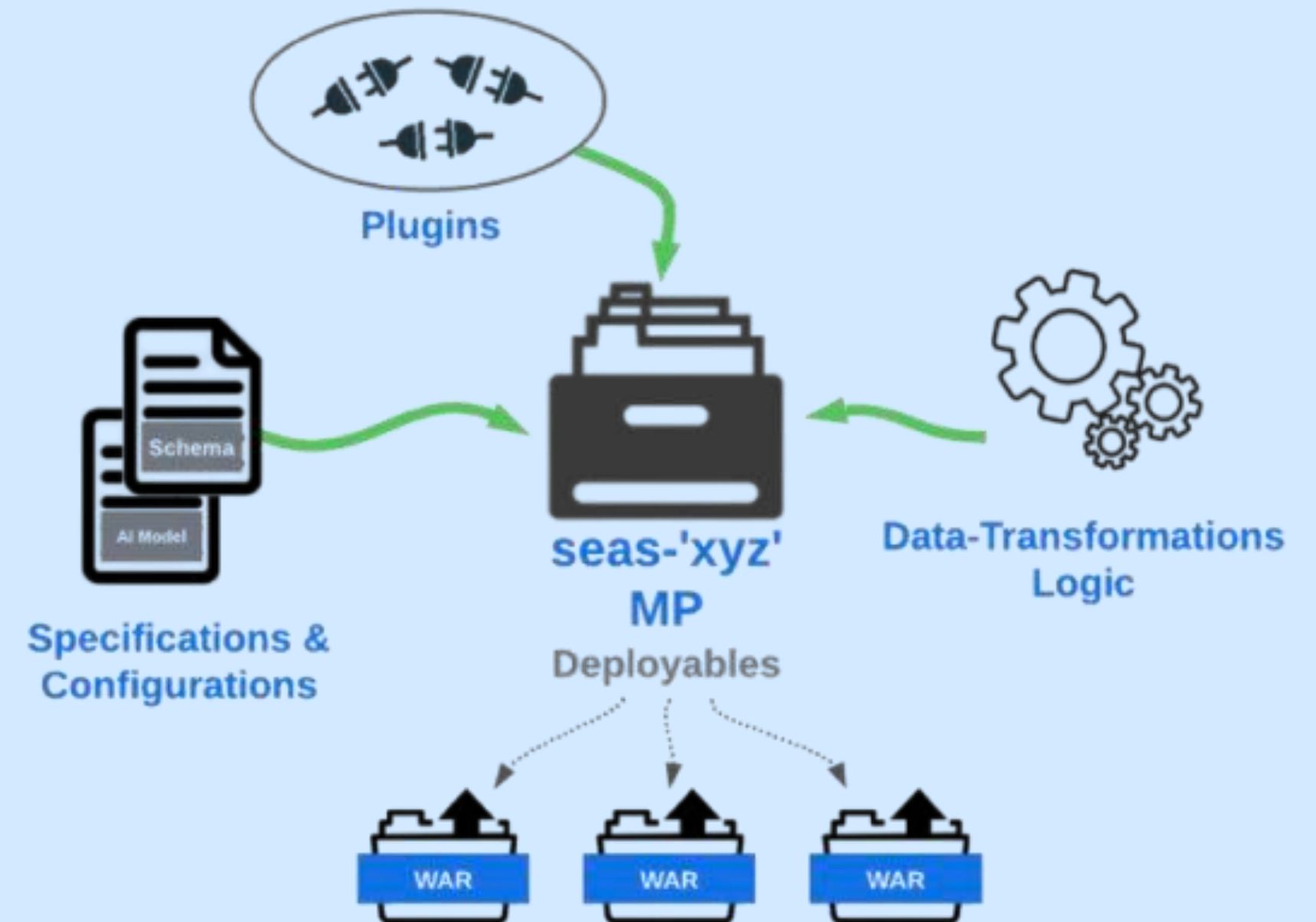
- Serving layer handles search queries and returns ranked results to users.
- Retrieves matching documents and applies relevance scoring.
- Optimized for low-latency, high-throughput

LinkedIn's Legacy Search Infrastructure (1)

- Built on **SeaS(Search As A Service)** and **Galene** libraries
 - **SeaS** handles lifecycle of services and their interactions across a search cluster
 - **Galene** manages low-level indexing and live data updates
- Search use cases are implemented as **SeaS verticals** (since 2014)
 - An MP is an independently releasable unit with deployables/libraries
 - Initialized using a template with pre-wired platform artifacts (e.g., SeaS, Galene)

LinkedIn's Legacy Search Infrastructure (2)

- Application teams own the setup of SeaS verticals
 - Why do different teams need their own search vertical?
- Create a **SeaS-'xyz' multiproduct (MP)** via platform-provided template
 - Template includes **SeaS, Galene**, and other core platform artifacts
- Add **custom artifacts** to the MP
 - Implement business-specific logic for **indexing, retrieval, scoring, ranking**
- Define data transformation logic
 - Transforms offline/nearline **source of truth (SOT)** data before indexing
- Build process assembles all artifacts into **per-use-case WAR files**
- Vertical is now a **custom search solution** tailored to application needs



Problems with Legacy Search Infrastructure

1

Operational Complexity & Ownership

- SeaS is not a managed service; owned and operated by app teams + SREs
- Involves complex stateful components and interactions, making operations hard

2

Tight Coupling of Platform and Application Logic

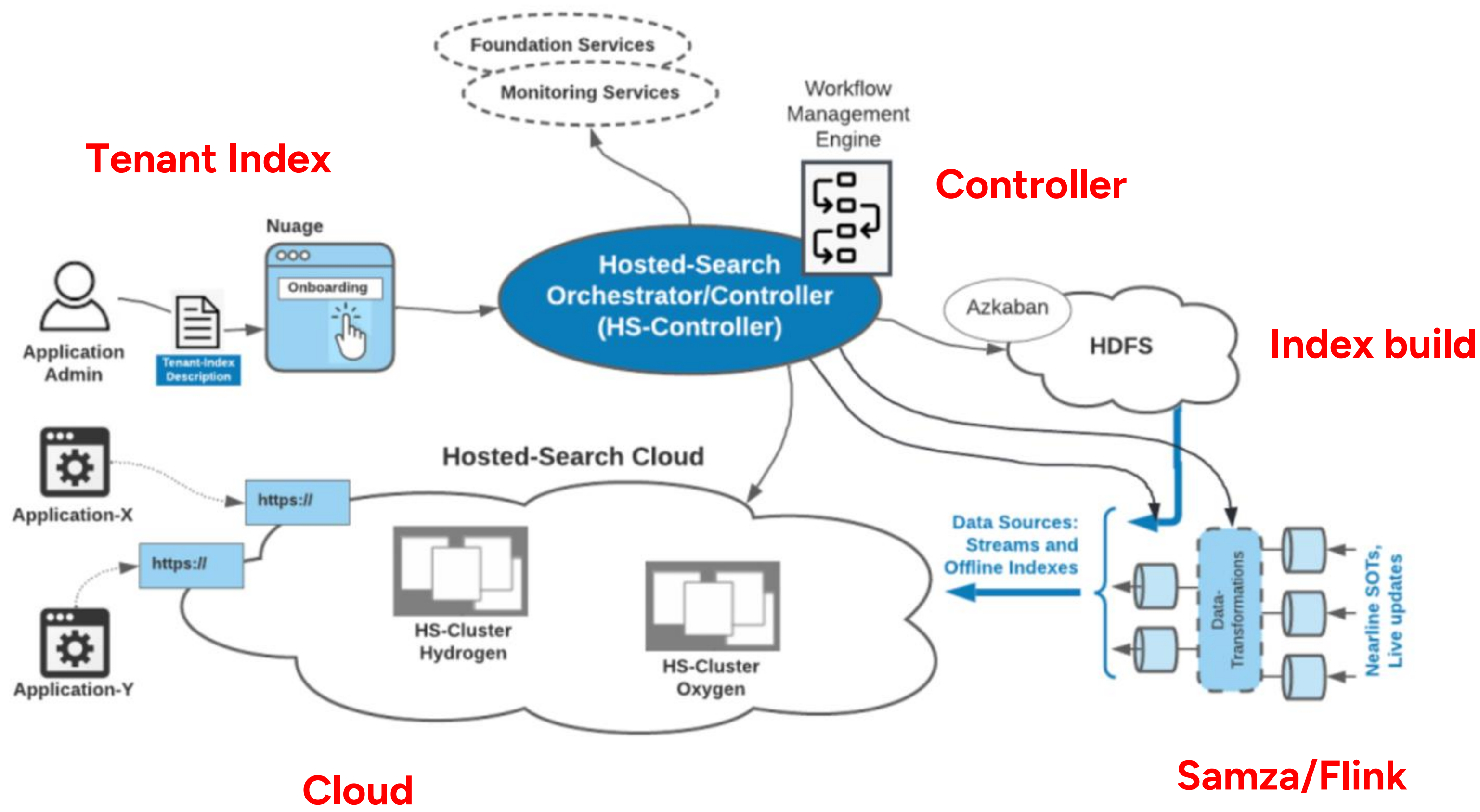
- App-specific logic is tightly coupled with platform code in the same MP
- This coupling makes development, debugging, and incident management difficult

3

High Cost, Expertise Requirement, and Scalability Limits

- Requires high search expertise and effort, which many app teams lack
- Not scalable for global search on Espresso tables, so never adopted for it

Hosted Search Architecture



Tenant Index Artifact MPs

Application Customization

Each Hosted-Search Tenant-Index (TI) allows app-specific plugins and configs

Custom logic for indexing, retrieval, scoring, ranking is defined by the application team

These artifacts are stored in a dedicated artifacts-MP for the TI

Artifacts-MP Structure

The artifacts-MP contains only logic/config code — no deployable WARs

Contrasts with SeaS verticals, where each MP builds its own WAR

Ensures clean separation of application logic from platform deployables

Deployment & Operations

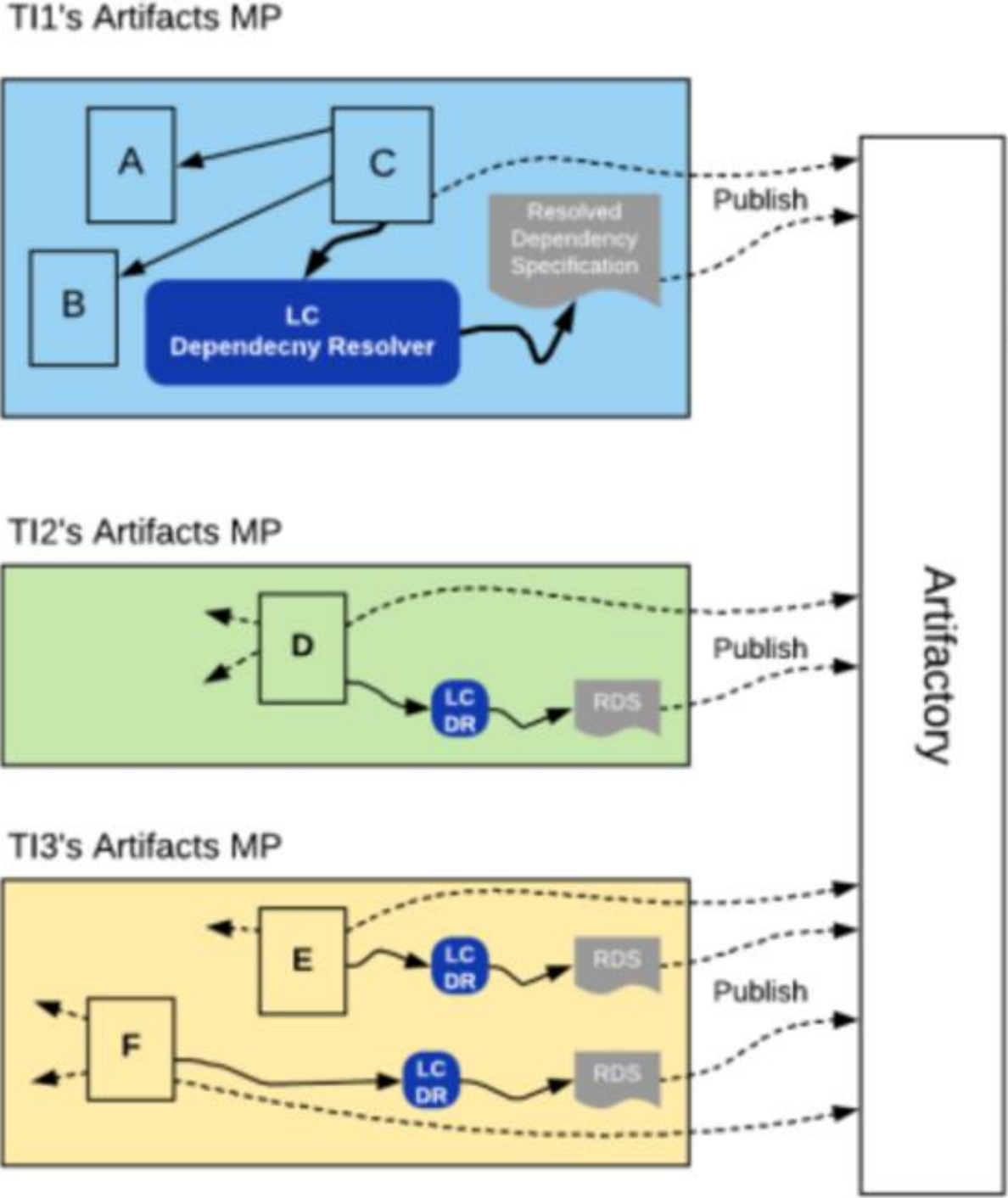
Only platform WARs are deployed in Hosted-Search

App-specific artifacts are fetched and injected at runtime via Layer Cake (LC)

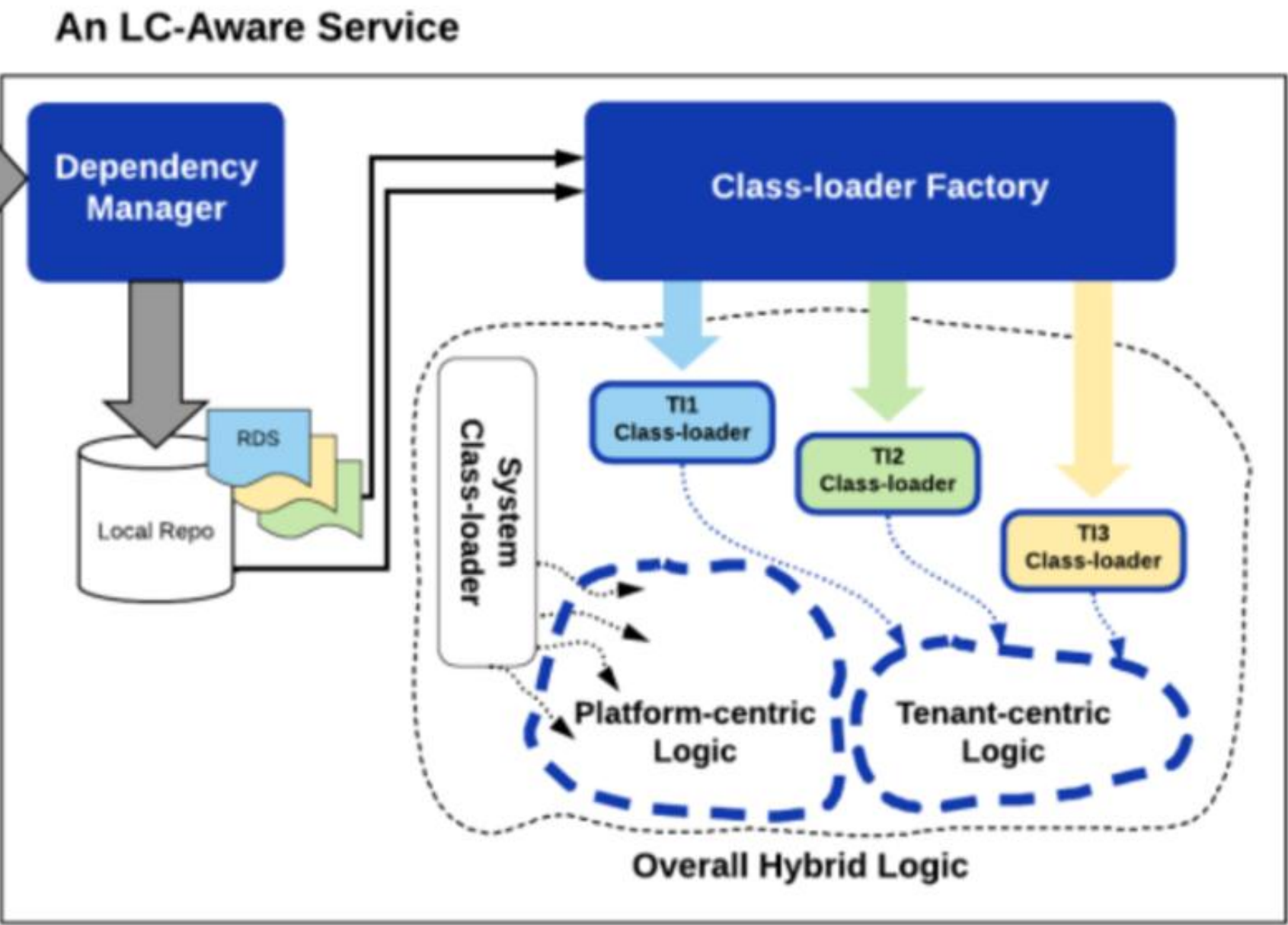
App teams do not handle site-up or operational responsibilities

Layer Cake Machinery

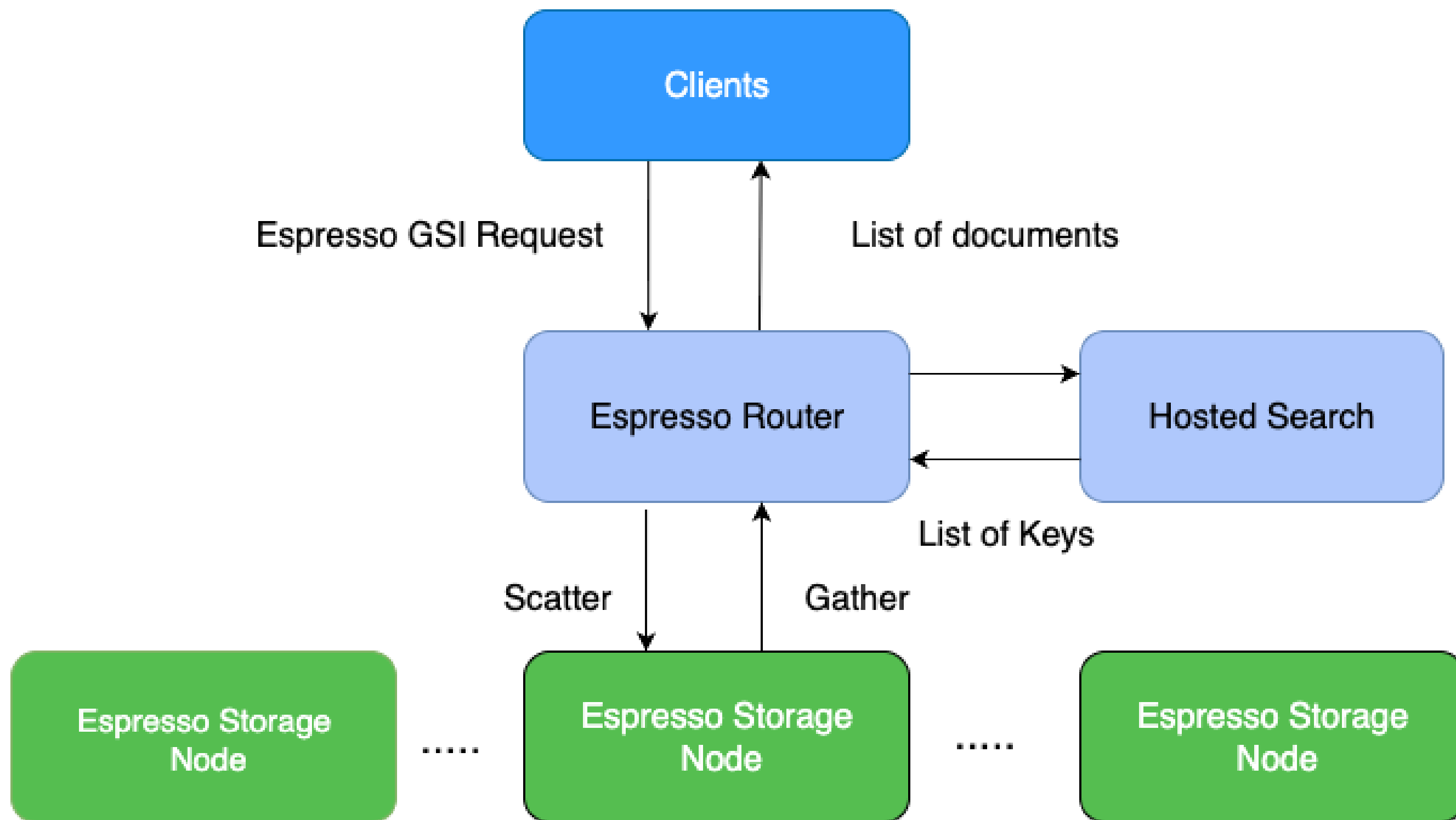
1. LC-DR & RDS



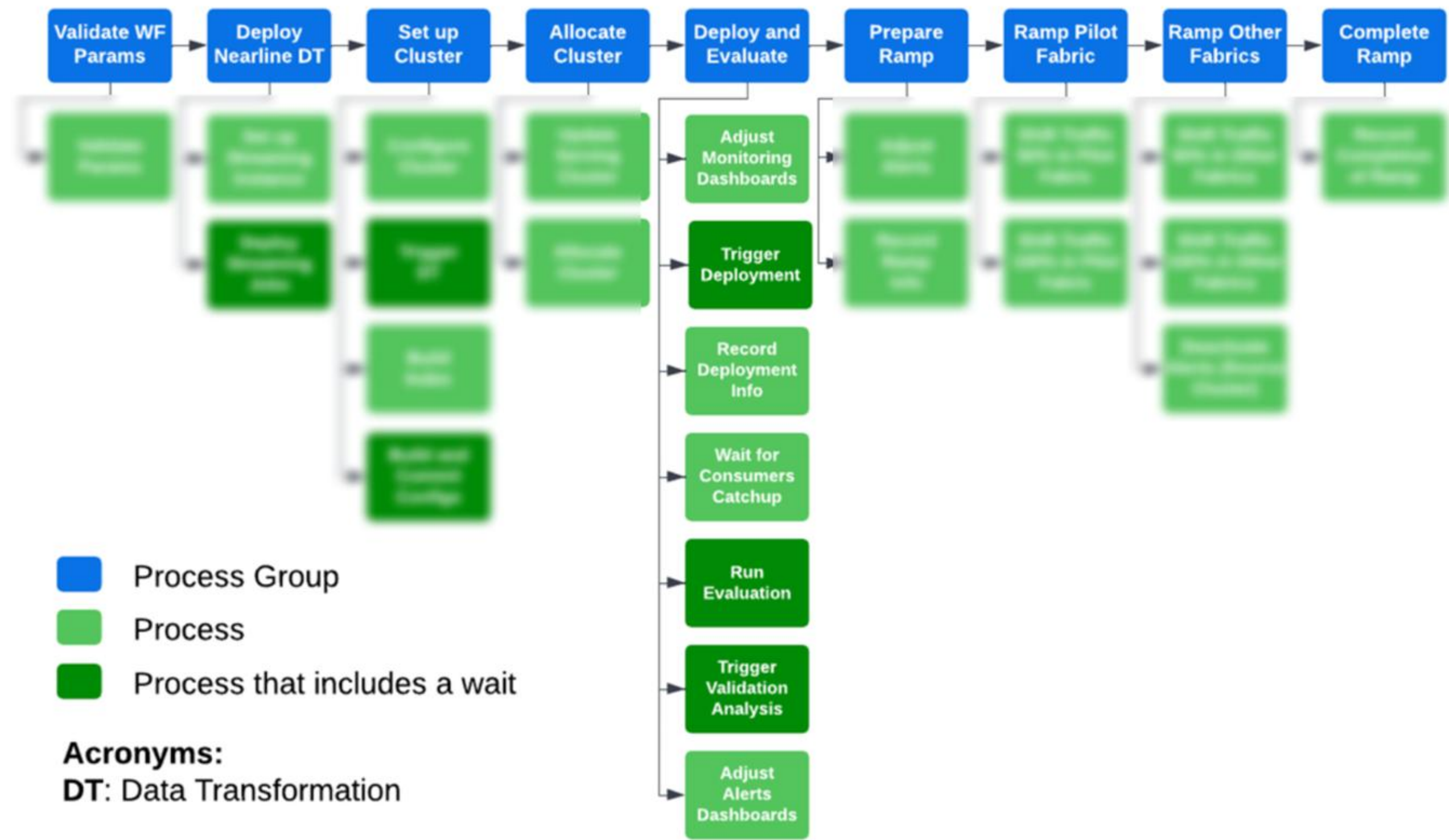
2. DM



Global Secondary Index in Espresso



Blue Green Workflows



**170+ Onboarded
tenants/use-cases
to Hosted Search**

Conclusion

- **Operational Overhead:**

- Hosted search is a fully managed platform with no setup or maintenance required from application teams.

- **Development Velocity:**

- Teams can focus on application logic and user experience, accelerating improvements on the search front.

- **Built-in Reliability & Compliance:**

- Automated handling of data freshness, security, and compliance reduces risk and ensures consistency

Thank You.

