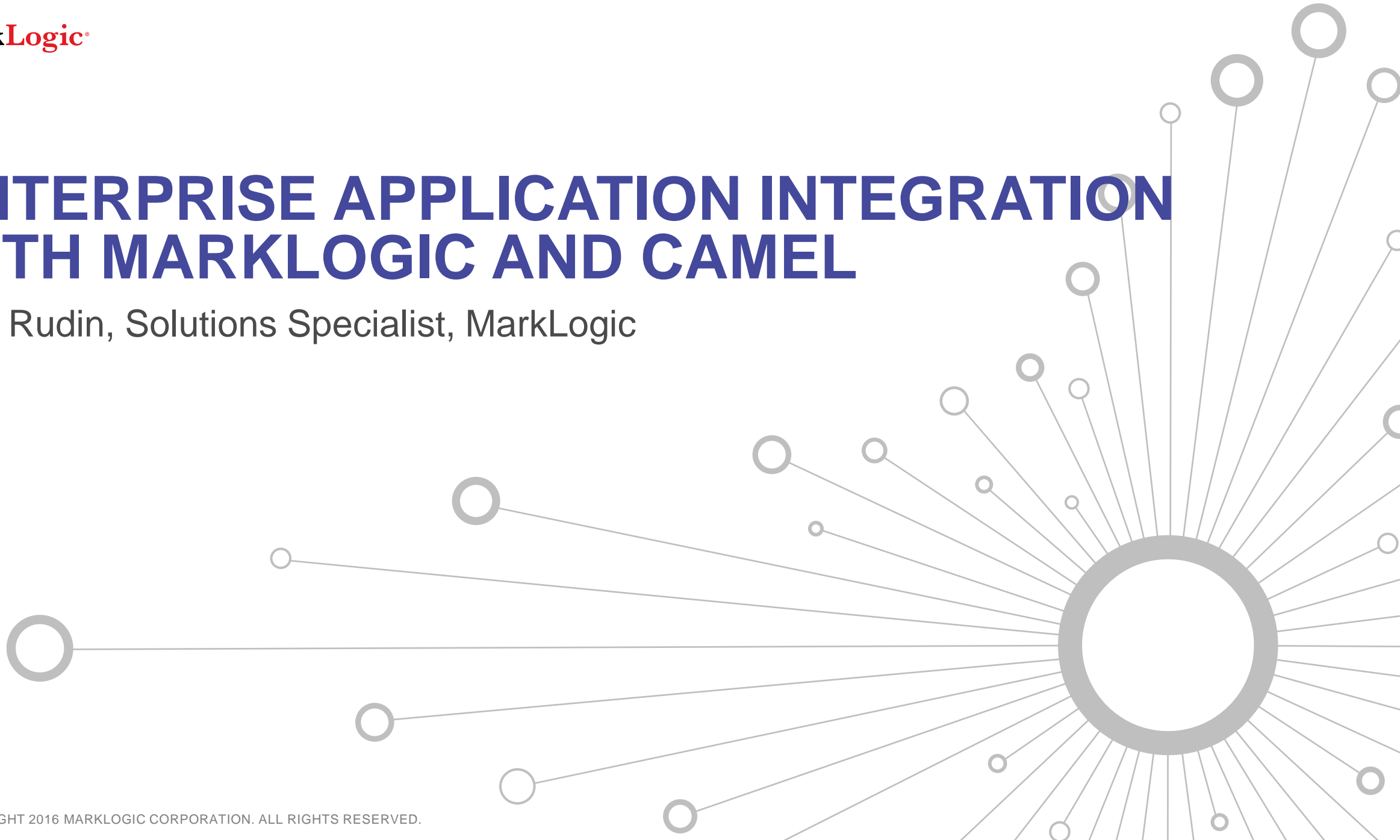


ENTERPRISE APPLICATION INTEGRATION WITH MARKLOGIC AND CAMEL

Rob Rudin, Solutions Specialist, MarkLogic



Agenda

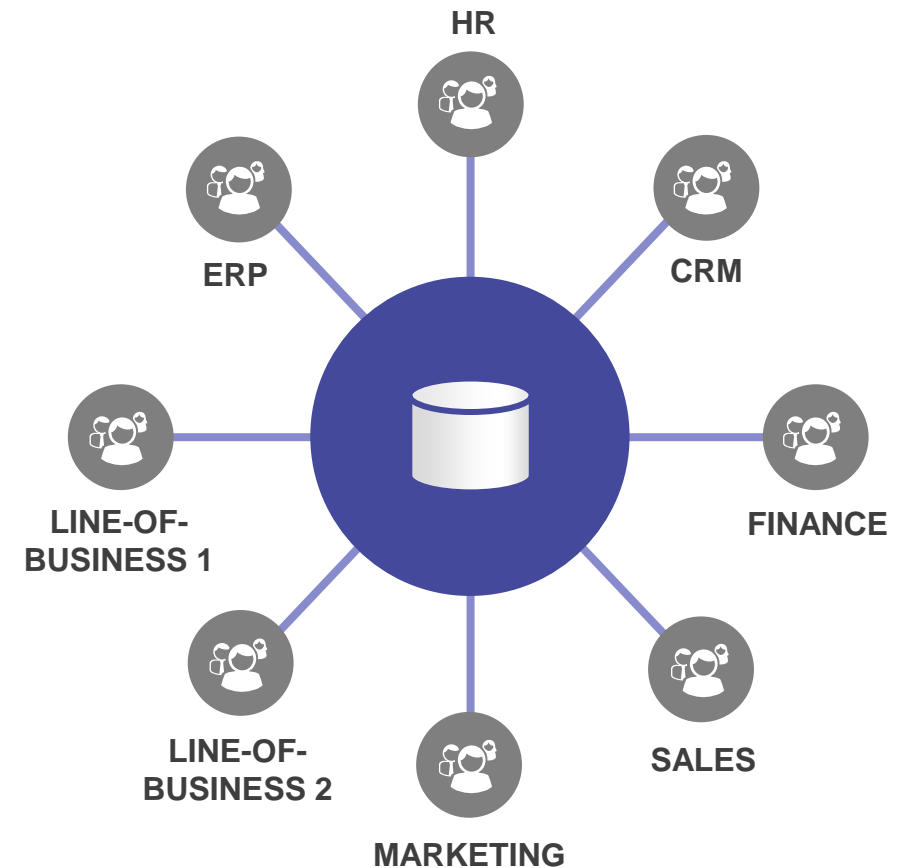
- Introduction
- Problems solved by EAI
- How EAI and MarkLogic work together

Introduction

- Rob Rudin
- Solutions Architect at MarkLogic, 2+ years
- Former MarkLogic customer, 5+ years
- <https://github.com/rjrudin>

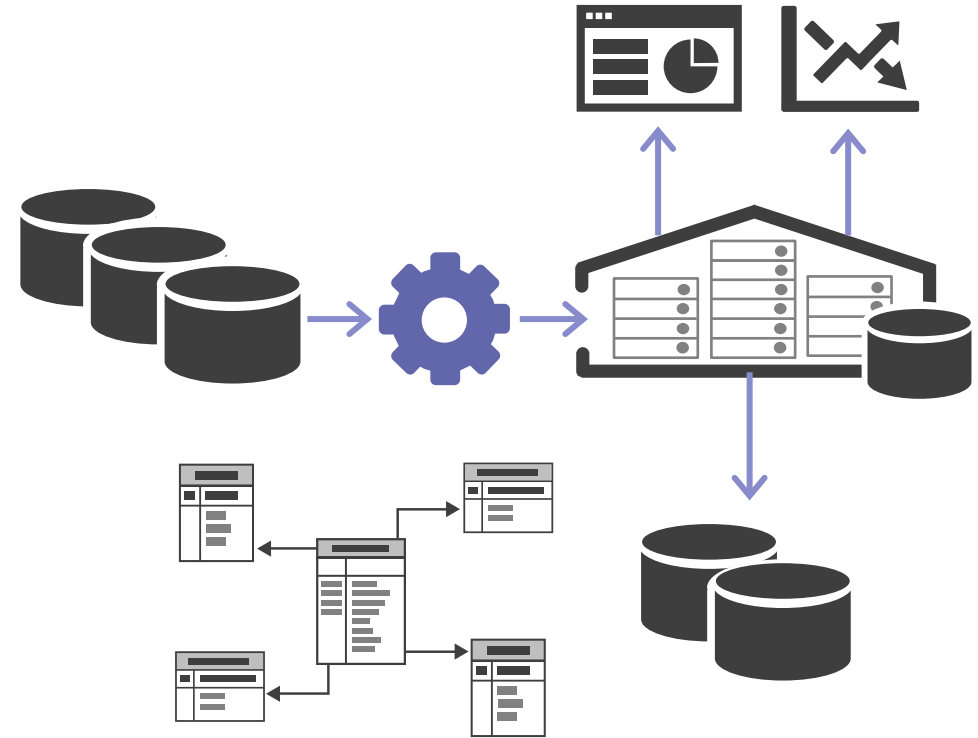
The Operational Data Hub

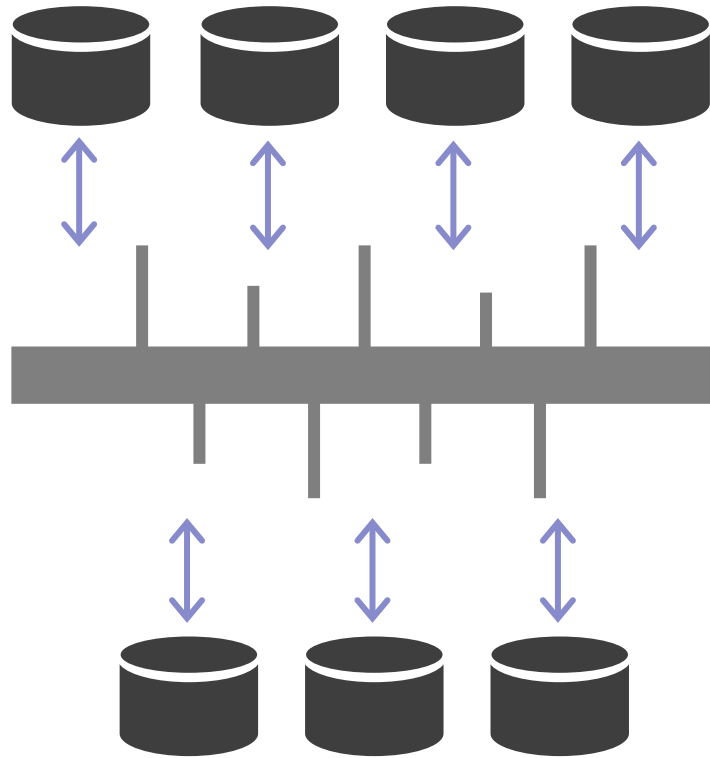
- A flexible, scalable operational database
- All types of data
- Convergence between analysis and operations
- **How do we keep these in sync?**



Real World Example

- JMS message queues
- Multiple relational databases
- Web services
- Exports to BI tools
- EJB (!) interfaces
- **MarkLogic brought into the middle**





Enterprise Application Integration (EAI)

- Emerged in various flavors from the late 90s into 2000s (point-to-point, SOA, ESB, etc.)
- Addresses integration for the **run-the-business** functions

EAI and ETL, in General Terms

EAI

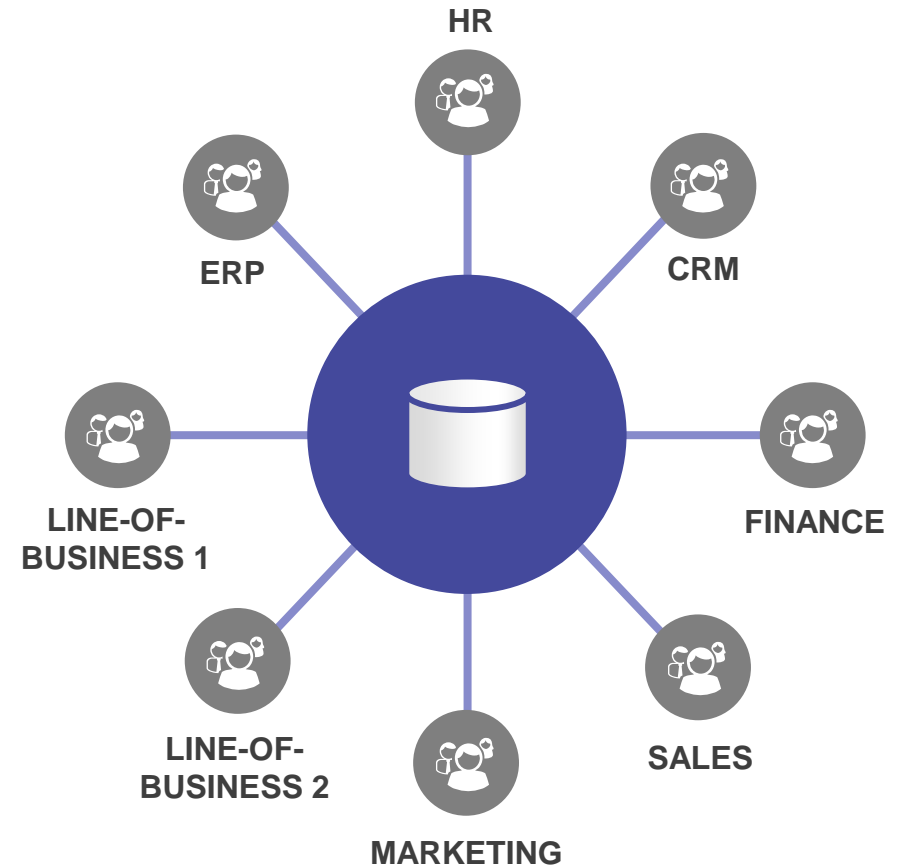
- **Frequent**
Many times a day
- **Message-oriented**
Messages on a queue, files in a directory
- **Many steps**
Parallel steps; point to point; broadcasting; aggregating
- **Many systems**
Databases, web services, external systems, command line interfaces

ETL

- **Infrequent**
Maybe just once
- **Batch-oriented**
Migrating a database, importing a data dump
- **Well-defined steps**
Extract, transform, load – transform could be complex
- **Database to database**
Not always, but typically

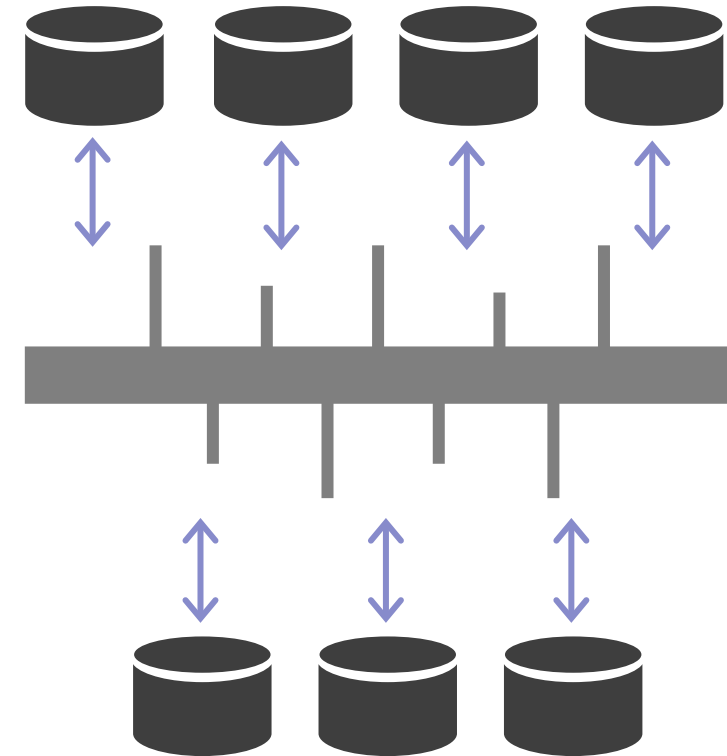
EAI and the Operational Data Hub

- How do we keep these in sync?
 - Data feeds
 - Data transformation
 - Entity resolution
- How can EAI help us?
 - And how do EAI and MarkLogic work together?



Better EAI with MarkLogic

- Understand what MarkLogic can do
- Understand what EAI tools can do
- Recommendations for blending the two together

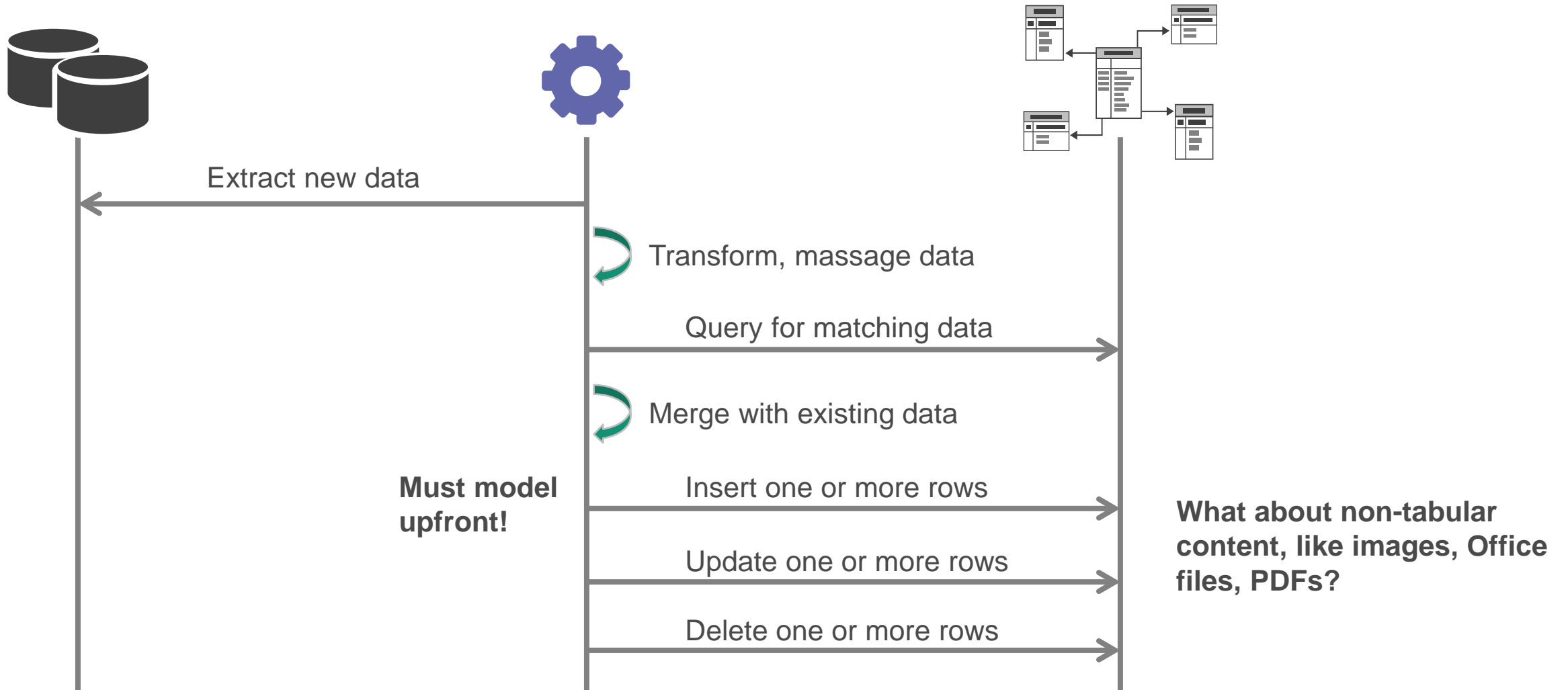


MarkLogic Changes the Equation

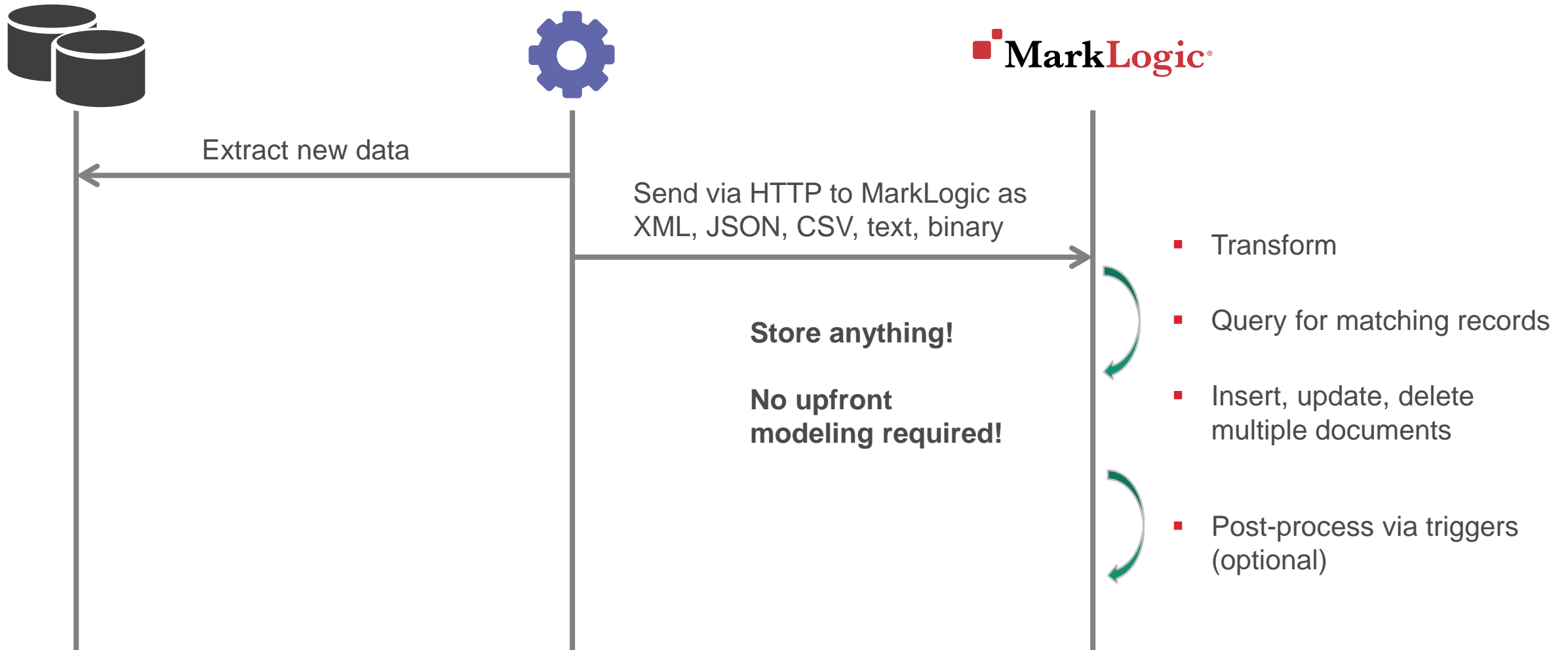
- Traditional approach – do everything in middleware
 - Message bus, ESB, app server
- Need to leverage MarkLogic's application services to maximize performance and simplicity



Traditional EAI



How EAI can Work with MarkLogic



MarkLogic Facilitates EAI

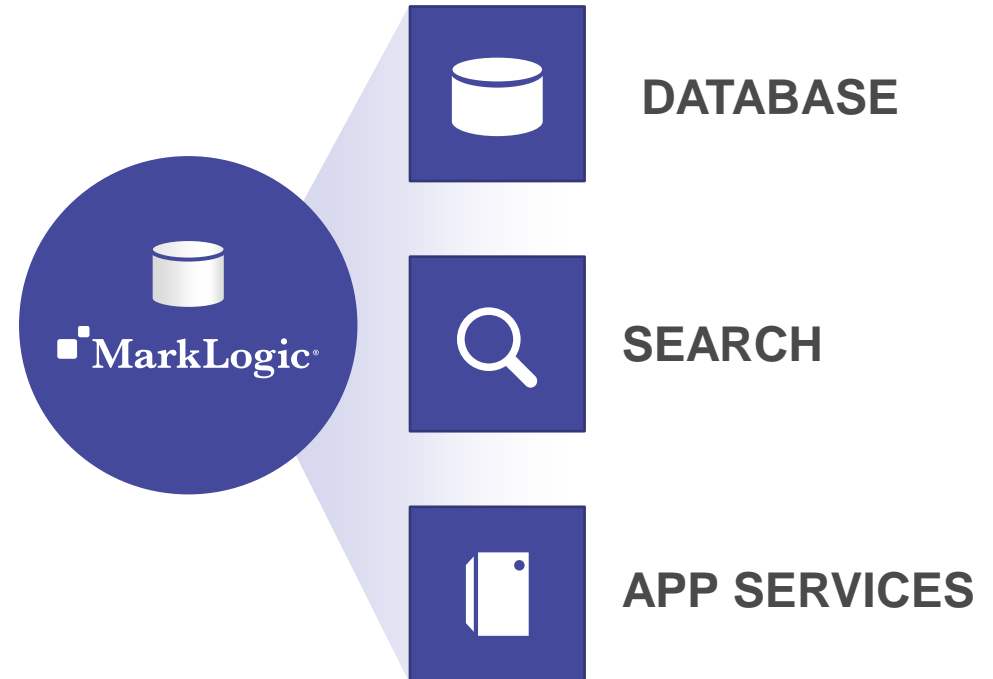
- Store any data – structured, unstructured, images, Office files, PDFs - anything
- REST API – easy to load documents with transforms
- Triggers – easy to add processing after data is loaded
- Indexes – easy to resolve entities
- Server-side programming – easy to call other web services, sends emails, etc

But Don't Get Rid of EAI

- Extraction – still need to integrate with a wide variety of data sources
- External services – still need to invoke a wide variety of external services
- Library reuse – may need to reuse libraries that can't run within MarkLogic
- Content routing – still need a way to all the steps in an integration

Important!

- MarkLogic can do a lot
 - It's a database
 - It's a search engine
 - It's an application layer
- But it's not an EAI tool
 - Use one that exists



Recommendations for EAI with MarkLogic


- We'll use 3 use cases for analyzing how to combine EAI with MarkLogic
 - A simple “hot” folder
 - A hot folder that depends on an external service
 - An ingestion pipeline with entity resolution and complex transformation rules
- We'll use Apache Camel as our EAI tool

Apache Camel


- Open source integration framework – based on Enterprise Integration Patterns
- Written in Java – integrates with dozens of other popular Java projects, as well as nearly any protocol you can think of
- Mature – first released in 2007



The Camel Ecosystem



Apache Camel Components



Category	Component	Description
Automating Tasks	Amazon	For working with Amazon's CloudWatch, S3, SNS, SQS, etc.
	Basic	For working with basic system tasks like file operations, etc.
	Batch	For working with batch processing tasks.
	Bean	For working with Java beans.
	Cache	For working with caching mechanisms.
	Command	For working with system commands.
	Consumer	For working with consumer tasks.
	Endpoint	For working with endpoints.
	File	For working with file operations.
	Producer	For working with producer tasks.
ESB	Activator	For activating/deactivating components.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Header	For setting/getting headers.
	Producer	For producing data.
	Queue	For using message queues.
	Router	For routing messages.
File I/O and Transfer	Amazon	For working with Amazon's S3, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Feeds	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Maintenance and Monitoring	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Persistence	Amazon	For working with Amazon's S3, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Special support	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Platform Support	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Testing	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Web Services	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Search Engines	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Networking	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Security	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Social Media	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
OSGi	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Java Message Service	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.
Endpoint Communications	Amazon	For working with Amazon's CloudWatch, etc.
	Batch	For working with batch processing.
	Bean	For working with Java beans.
	Cache	For caching data.
	Command	For executing system commands.
	Consumer	For consuming data.
	Endpoint	For connecting to endpoints.
	File	For reading/writing files.
	Producer	For producing data.
	Queue	For using message queues.

Apache Camel 2.12.1 and external components

Gliesian, LLC

<http://camel.apache.org/components.html>

Notable Camel Components

- File – watch for files to arrive in a directory
- HTTP – invoke HTTP endpoints
- Bean – invoke custom Java code
 - Great for integration with MarkLogic, as we're deeply invested in the Java ecosystem
- Broadcast – send a message to multiple components for parallel processing
- Dozens more...

Simple Hot Folder

- Use case – as files are created by other applications, copy them to a directory, and load them into MarkLogic
- Camel makes watching a directory easy - <http://camel.apache.org/file2.html>
- But how we do get a file into MarkLogic?

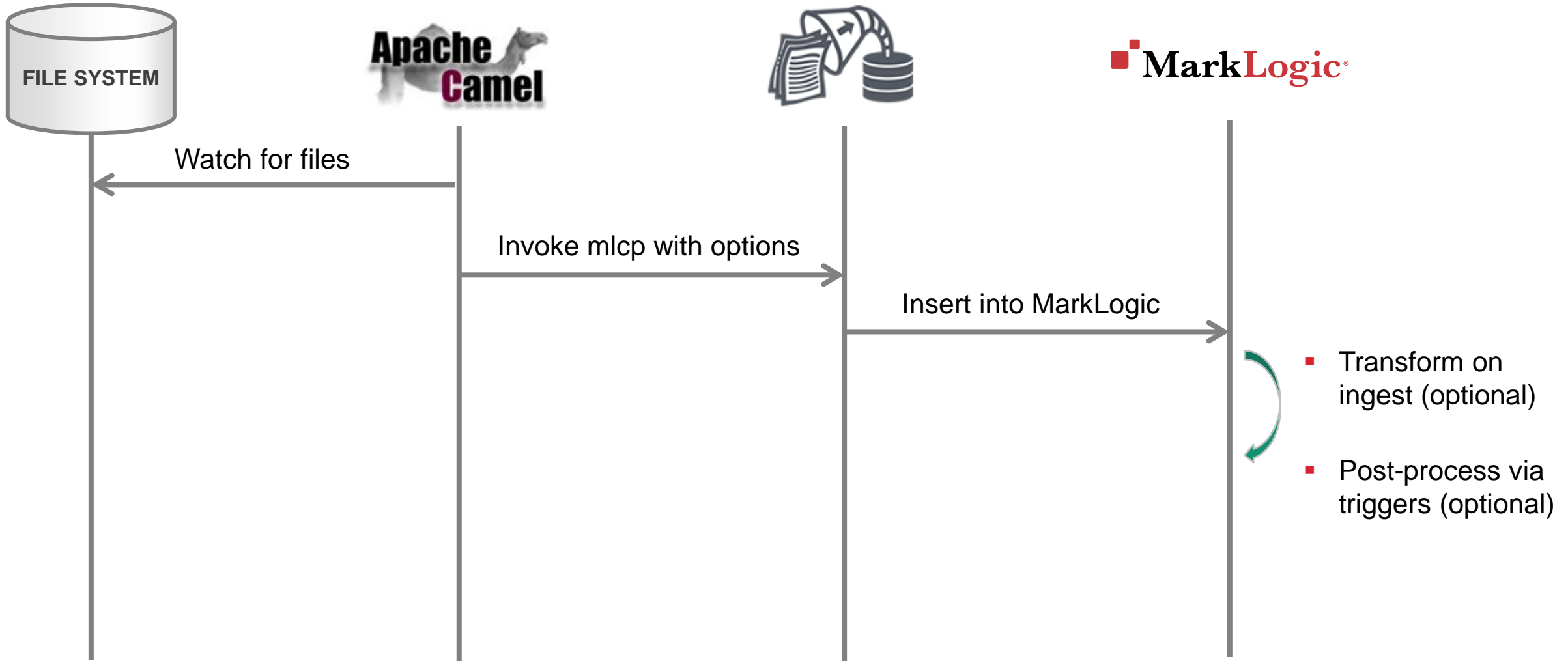
Hot Folder Options

- Could do nearly everything in Camel/Java
 - Can unzip - <http://camel.apache.org/zip-file-dataformat.html>
 - Or break apart a big XML - <http://camel.apache.org/splitter.html>
 - Can transform XML files - <http://camel.apache.org/xslt.html>
- Can then load via MarkLogic REST API
 - <http://docs.marklogic.com/REST/PUT/v1/documents>
 - Can set collections, permissions
 - Need a way to determine a URI

Hot Folder and Content Pump

- OR – use MarkLogic Content Pump (mlcp)
 - <https://docs.marklogic.com/guide/mlcp>
 - mlcp excels at loading from the filesystem
 - Can specify a transform, split up an XML file, split up a zip, generate a URI via an algorithm, spread load across the cluster, parse delimited files, load triples, and more
- So Camel handles watching the directory
- And mlcp + MarkLogic handles loading and indexing data

Camel and Content Pump



Hot folder and a Camel Component

- Camel can be extended via custom components
- mlcp is Java, so easy to integrate
- Demo
 - For more advanced usage, could route files to different mlcp endpoints based on file type, file name, etc. – all a good use case for Camel
- Recommendation – when loading data from a file system, use mlcp

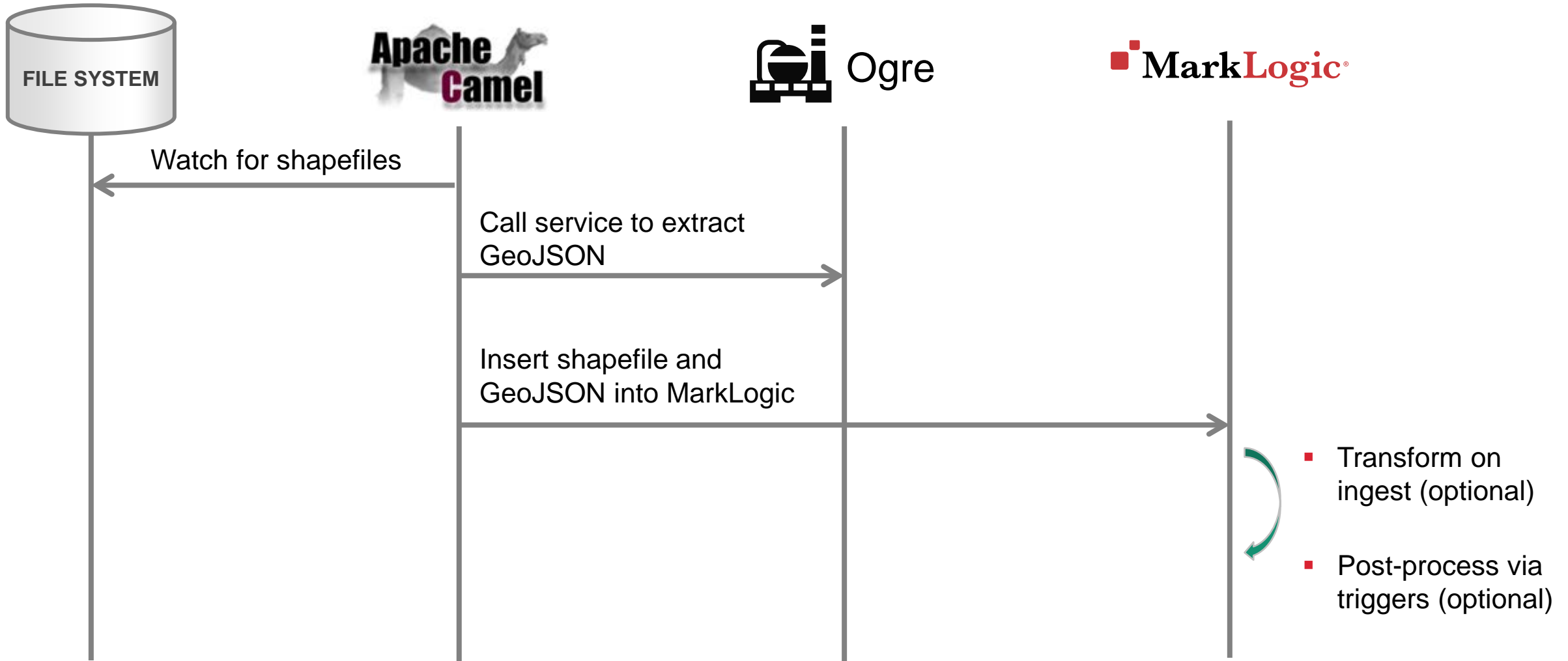
Hot folder with External Service

- Shapefiles – an Esri geospatial data format – are created by an application and copied to a directory
- For each file, extract the geospatial data as GeoJSON via an external service
 - OGRE - <http://ogre.adc4gis.com/>
- Then ingest both the shapefile and GeoJSON document into MarkLogic
 - Goal: searchable shapefile repository with original shapefiles

External Service Integration Options

- Could invoke Ogre in MarkLogic server-side code
- But there's a command line Ogre client too
- Consider:
 - Testability – easier to mock calls to Ogre in Java code in Camel
 - Availability – may need to support both web Ogre and client-side Ogre

Camel and an External Service



Camel and an External Service

- Consider making a custom Camel component
 - Support both web and command line OGRE
 - Reuse across Camel flows
- Demo
- Recommendation – in general, prefer invoking external services from EAI tool

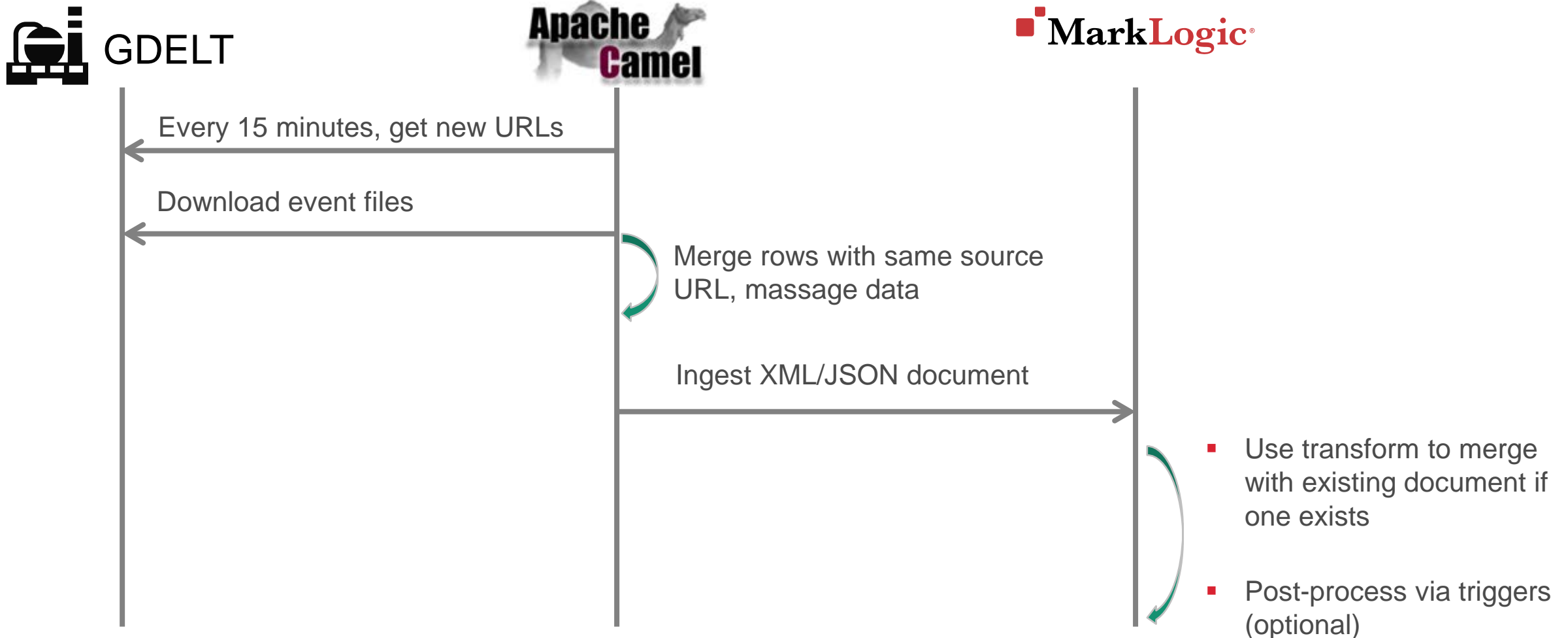
Complex Ingestion Pipeline

- GDELT Event Database
 - <http://gdeltproject.org/> and <http://data.gdeltproject.org/gdeltv2/lastupdate.txt>
- Contains news events with extracted persons, organizations, locations, and themes
- 7-page cookbook to understand the data format
 - http://data.gdeltproject.org/documentation/GDELT-Data_Format_Codebook.pdf

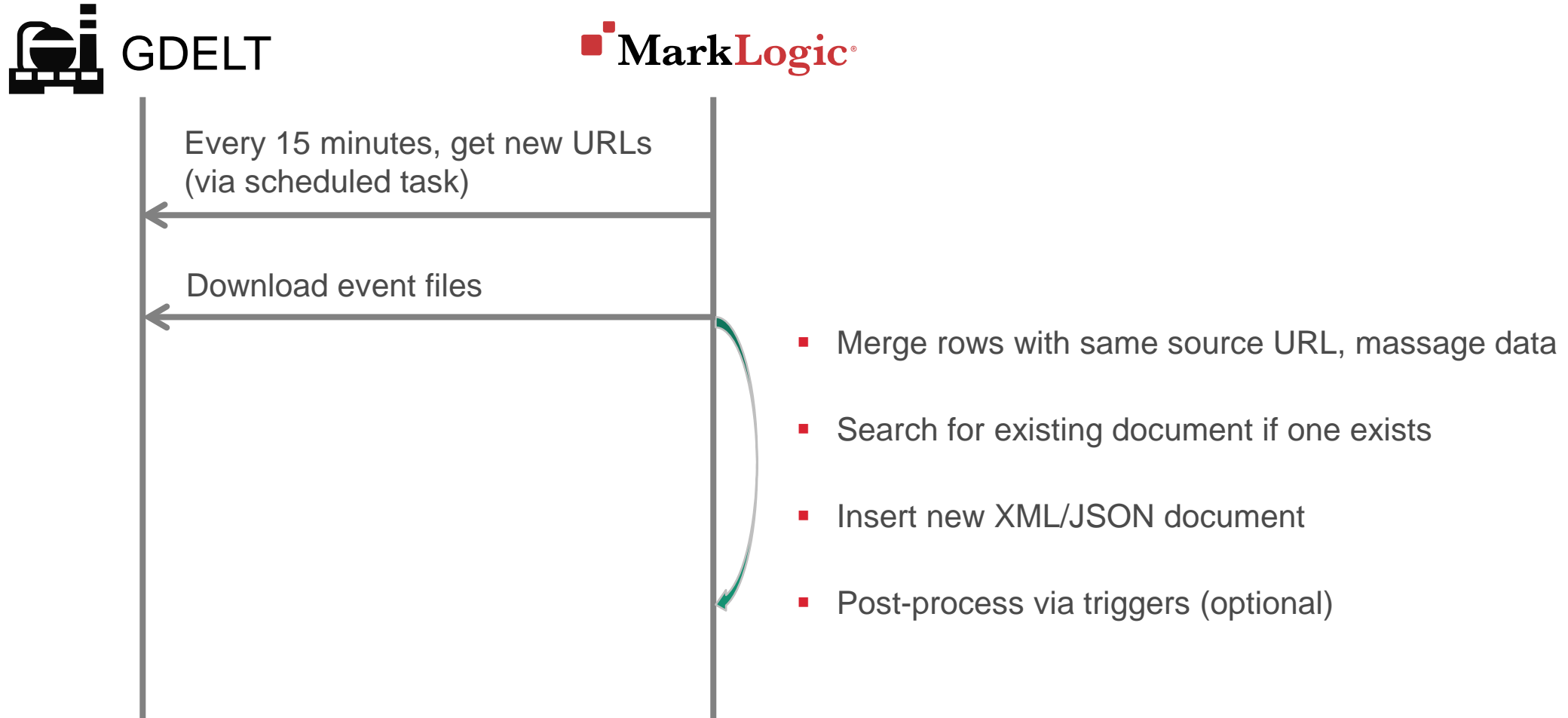
GDELT challenges

- External service
 - One call to get URL of new batch of events, another call to download batch
- Multiple lines per source URL
 - We want to consolidate those into a single document in MarkLogic
- URL may have been processed in previous batch
 - Will need to query MarkLogic to see if document already exists

How EAI can Work with MarkLogic



Could Also Do This All in MarkLogic Too



Benefits of Camel + MarkLogic

- Very large GDELT responses can be handled more easily
 - Camel can split this and parallelize processing
 - MarkLogic solutions exist (e.g. taskbot), though not CPF-based
- External service can be mocked for testing
- Developers may prefer complex logic to be in EAI language
- Demo

Complex Ingestion Recommendations

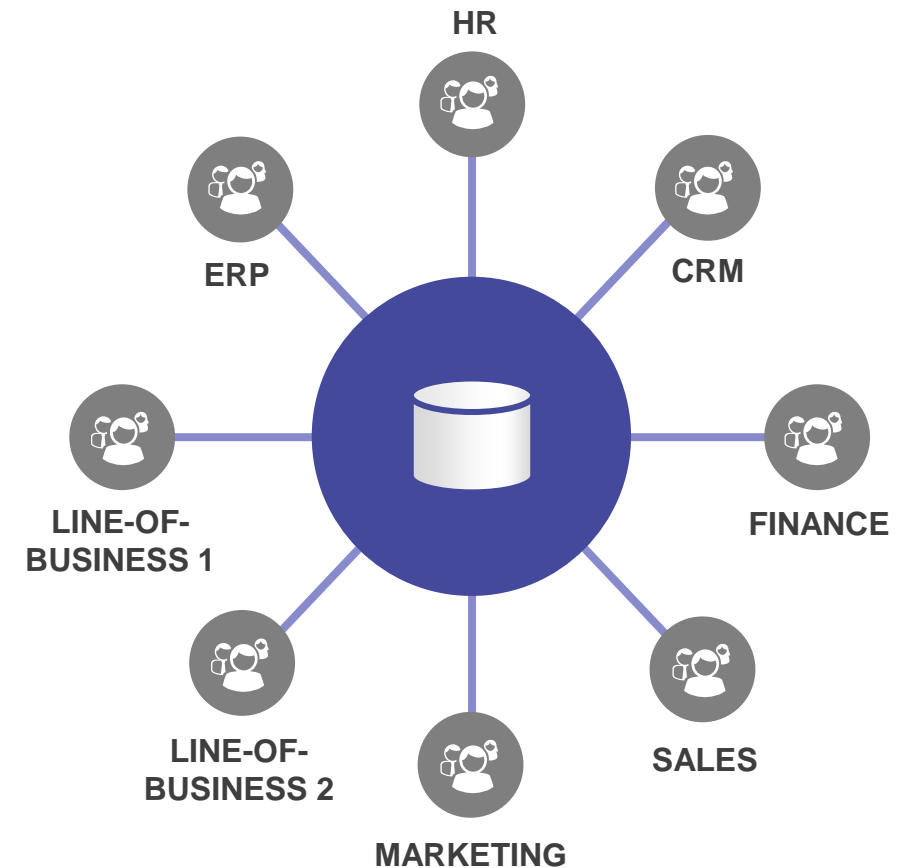
- Let EAI tool handle large payloads
 - Can leverage EAI patterns – routing, splitting, aggregating, etc.
- Consider need to mock external services for testing
- Consider personnel

Summary of Demos

- Simple hot folder
 - Recommendation – use Content Pump for ingesting files
- Hot folder + external service
 - Recommendation – prefer invoking external services from EAI tool
- Complex ingestion
 - Recommendation – let EAI tool manage large payloads
 - Recommendation – consider personnel when deciding where to implement

EAI and MarkLogic and the Operational Data Hub

- Understand what MarkLogic can do
- Understand what EAI can do
- Get and follow recommendations
 - MarkLogic Consulting Services
 - marklogic.com/training
 - stackoverflow.com/questions/tagged/marklogic



Q&A