



Technical Deep Dive of the AEM 6 Platform

Jukka Zitting | Senior Developer



Project Oak

Part of Apache Jackrabbit

- fully open source, ALv2
- <http://jackrabbit.apache.org/oak/>
- one of the most active Apache projects by commit counts

Fresh implementation of JCR 2.0

- mostly backwards compatible
- some optional features excluded to avoid bad performance/scalability tradeoffs

Project timeline:

- 2008: initial design ideas
- 2011: prototyping
- 2012: project launched
- 2014: Oak 1.0 released

Scalability

- Large repositories
- Distributed repositories

Throughput

- Improved performance
- Improved concurrency

Features

- Flat hierarchies
- Complex ACLs

Flexibility

- Pluggable components
- OSGi-friendly

Key differences

Oak

- MVCC
- tree persistence
- designed for scalability
- plugin architecture
- pluggable query indices

CRX2

- synchronous updates
- key-value persistence
- clustering as add-on
- static extension points
- one index per workspace

Architecture

JCR

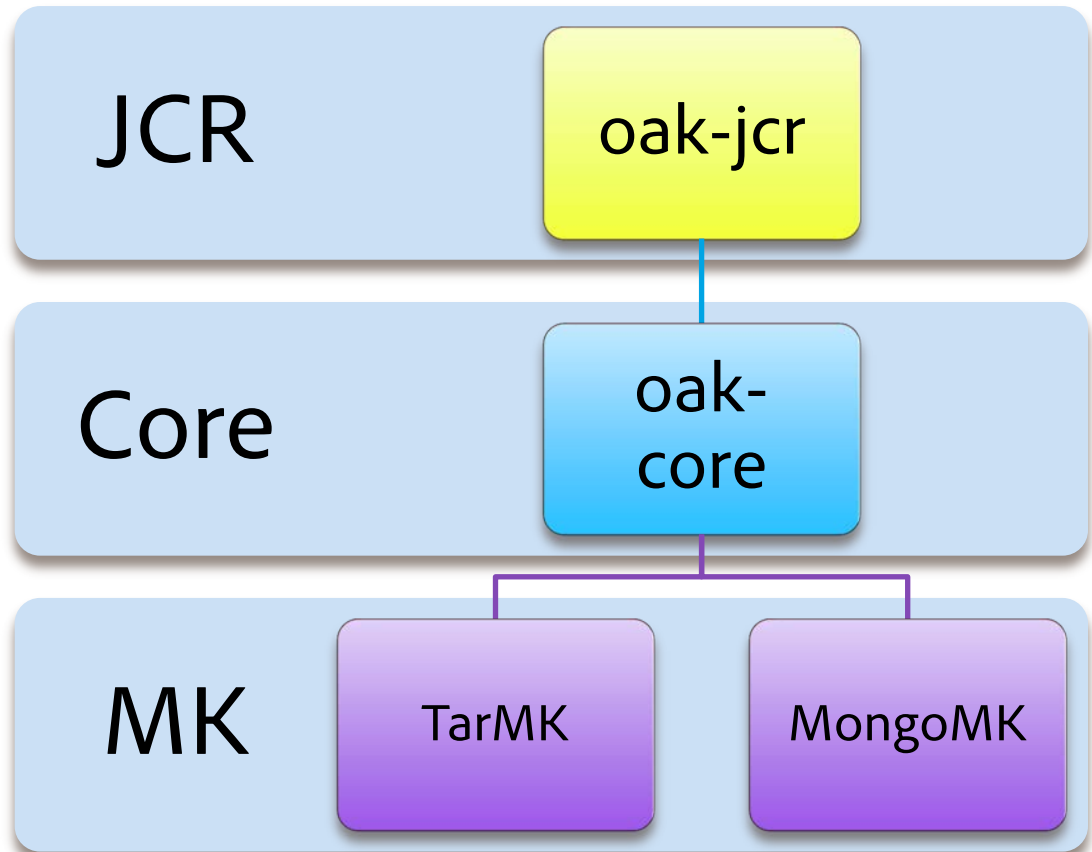
- JCR API binding
- convenience and safety features like auto-refresh and thread-safety guards

Core

- high-level functionality
- search, versioning, security, etc.
- most features implemented as pluggable extensions

MicroKernel

- versioned tree storage
- clustering, caching, etc.



For more details, see:

<http://www.slideshare.net/jukka/oak-the-architecture-of-apache-jackrabbit-3>



Deployment scenarios



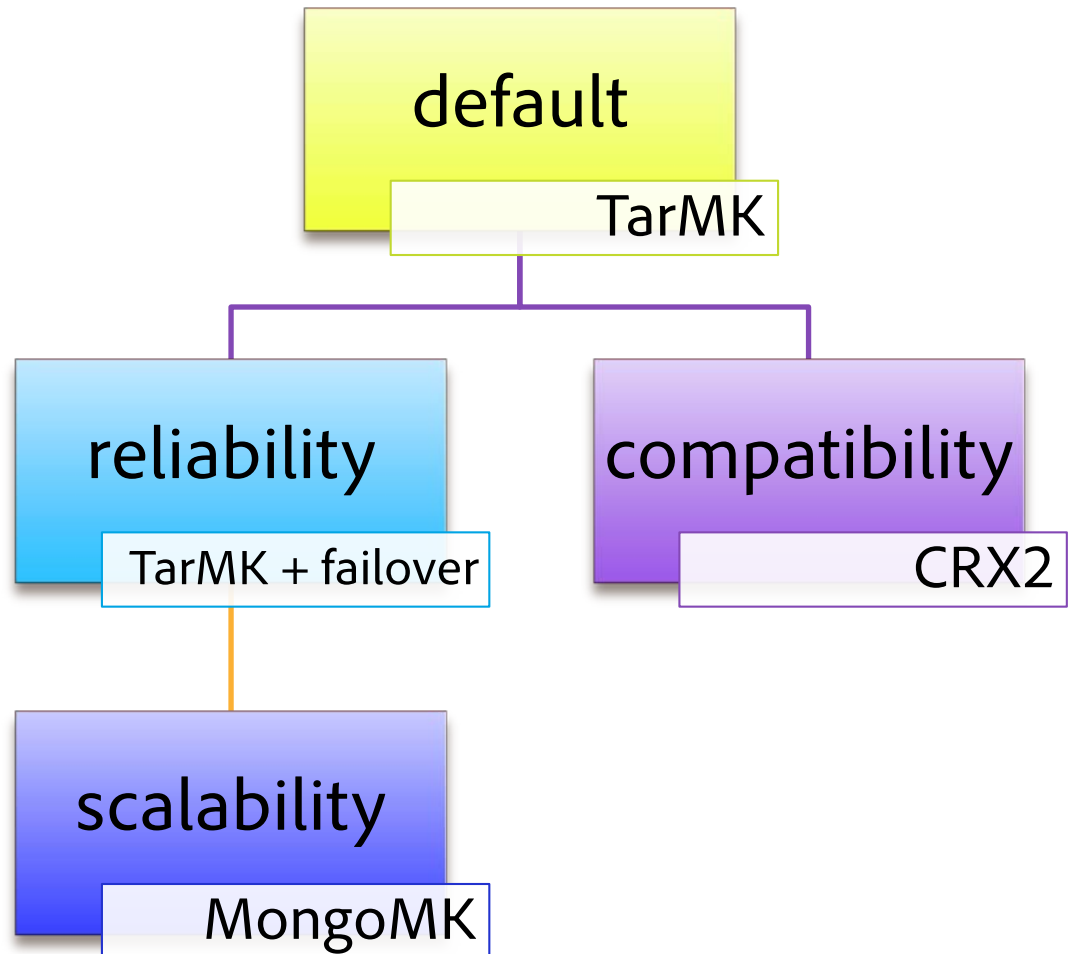
Which repository backend to use for AEM 6.0 author deployments?

Instructions:

- start from the top
- follow lines to add required features
- stop when happy

Note:

- each step adds complexity
- most steps can be postponed to when actually needed



AEM6 Publish

Which repository backend to use for AEM 6.0 publish deployments?

Instructions:

- pick your main use case

Note:

- in most cases it will be possible to switch the backend later on, though the migration may be a bit costly

Publishing

- TarMK
- farm of replicas

User generated content

- MongoMK
- publish cluster

Simple default deployment option

Heavily optimized for
single-node performance

- memory mapping (64bit JVMs)
- compactness
- locality of reference

Optionally with a data store
for large binaries

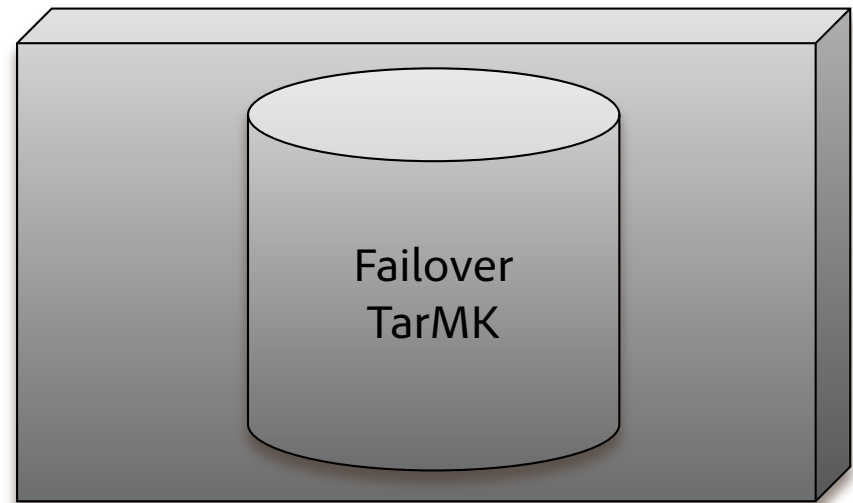
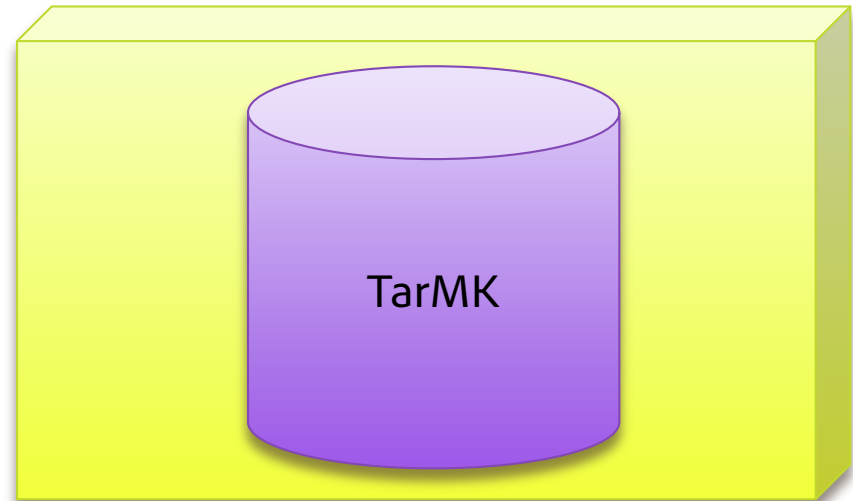
- migration from crx2
- external storage (NAS/SAN, S3)



TarMK + failover

For added reliability

- continuous, incremental backup to a separate failover server
- requires an external smart firewall, load balancer or monitoring tool (Nagios, etc.) for triggering the failover

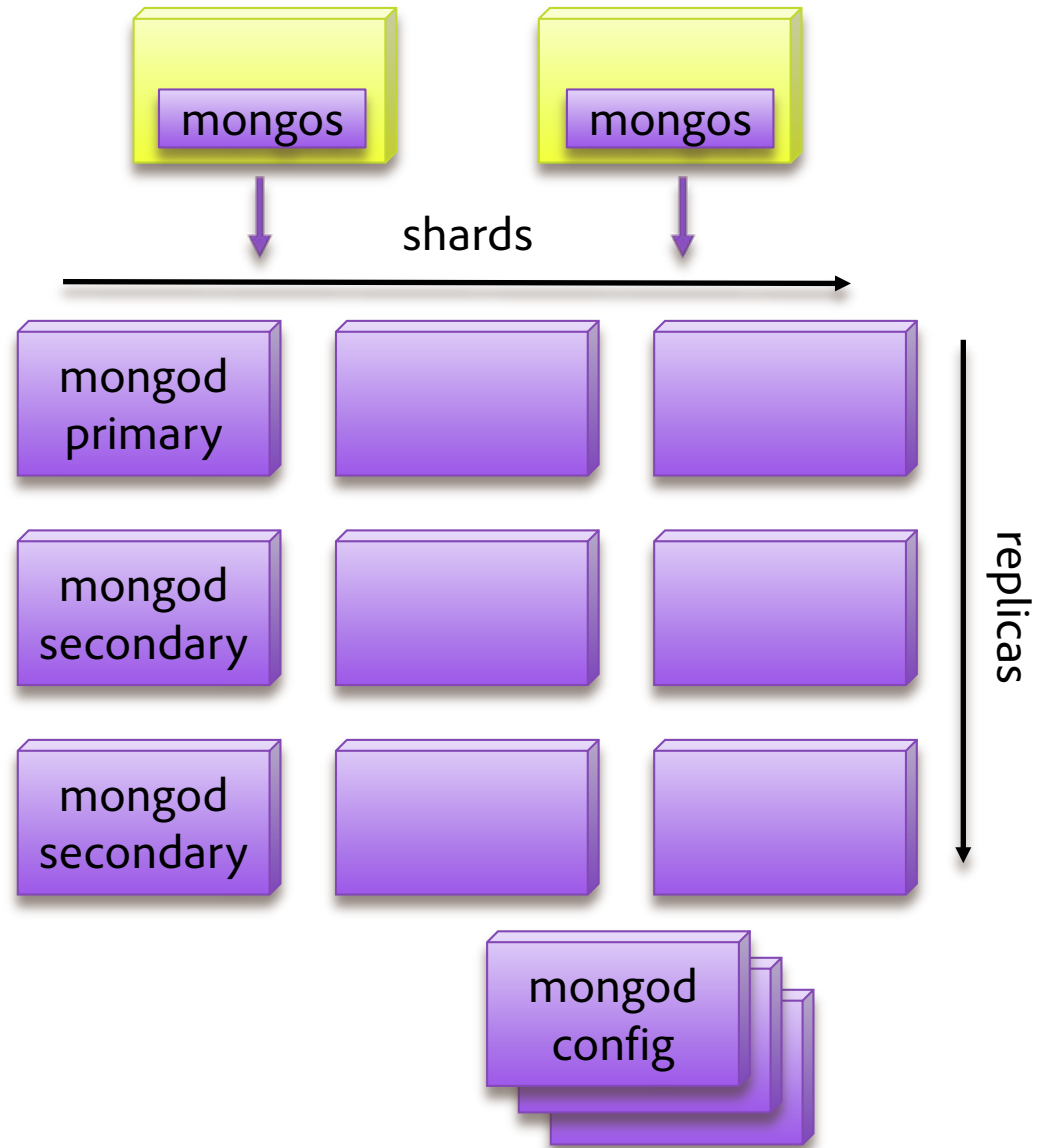


MongoMK

Reliability + scalability

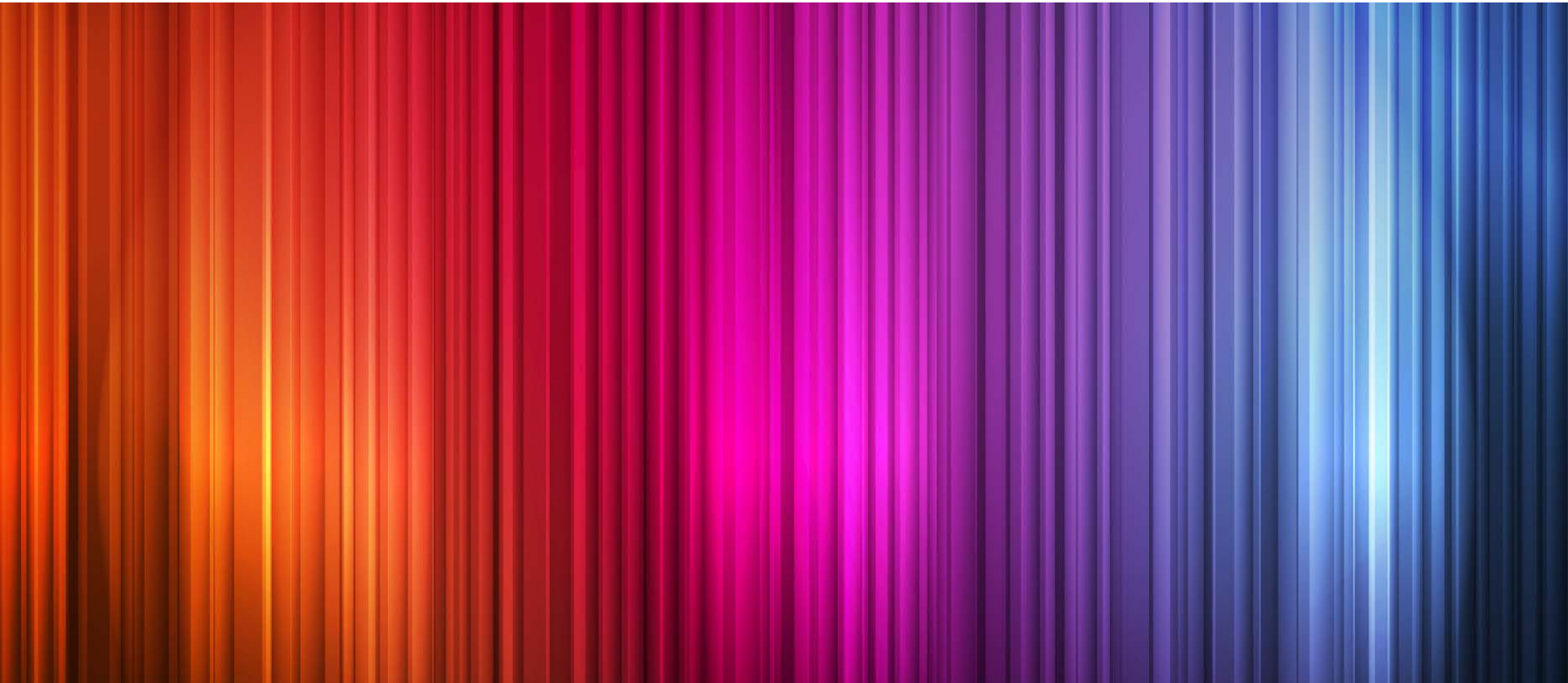
Any number of AEM6 cluster nodes can share an underlying MongoDB cluster

- standard MongoDB replication and sharding features
- all shared state in MongoDB
- eventually consistent





Content Migration



Content Migration

How to migrate an existing
AEM 5.x deployment
to AEM 6.0 with Oak?

BTW, why the migration?

- radically different storage format
- need for an explicit decision because of changes in backwards compatibility

For details, see:

<http://docs.adobe.com/content/docs/en/aem/6-0/deploy/upgrade.html>

Upgrade to AEM6 with CRX2

see upgrade instructions



Do a full backup

see backup instructions



Migrate from CRX2 to Oak

see next slide

Content Migration

How to migrate a CRX2 repository to CRX3 with TarMK?

Prerequisites:

- use Java 7 or higher
- first upgrade to AEM 6

For MongoMK:

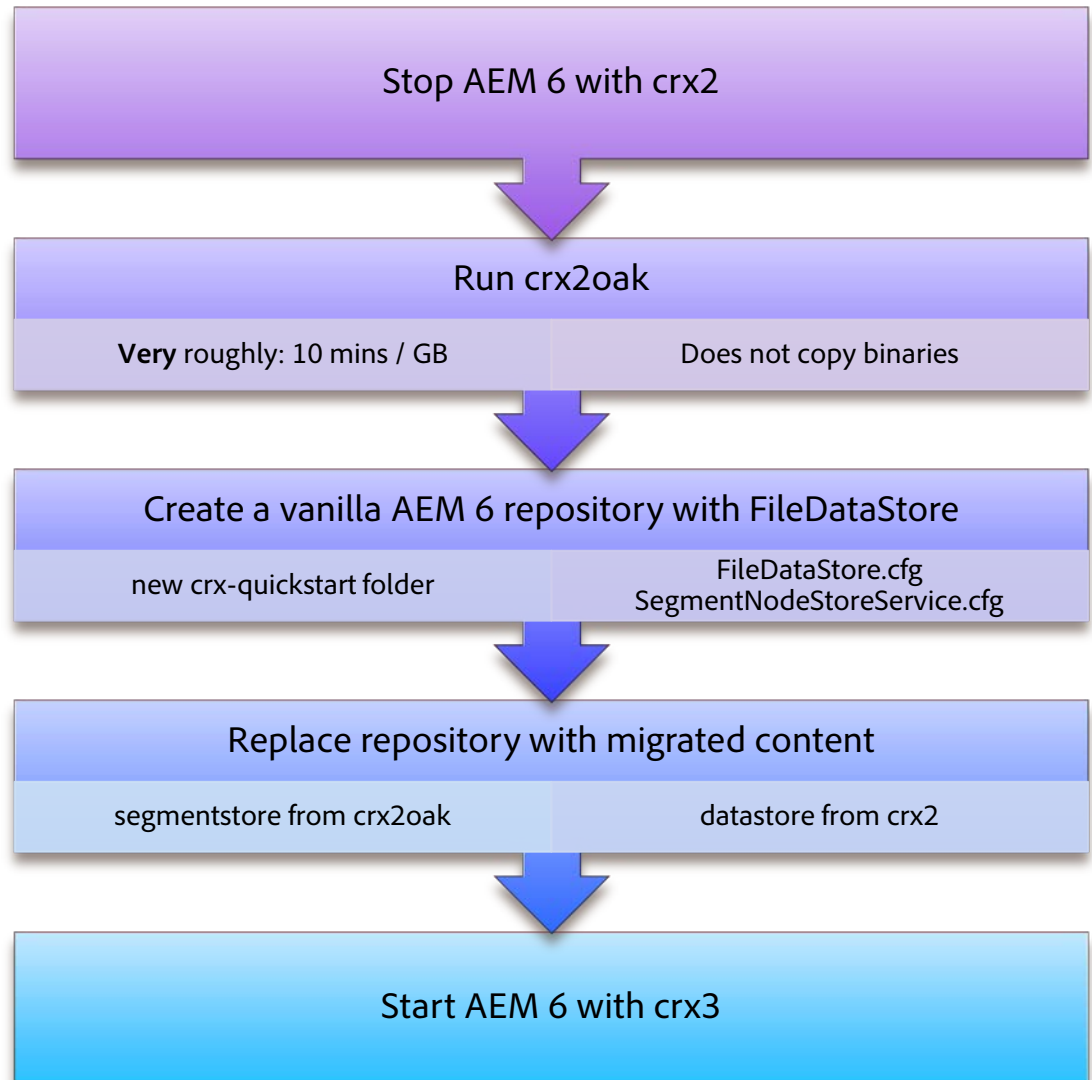
- -r crx3,crx3mongo
- -Doak.mongo.uri
- DocumentNodeStoreService.cfg
- see documentation for details

For custom data stores:

- see documentation for details

For details, see:

<http://docs.adobe.com/content/docs/en/aem/6-0/deploy/upgrade.html>

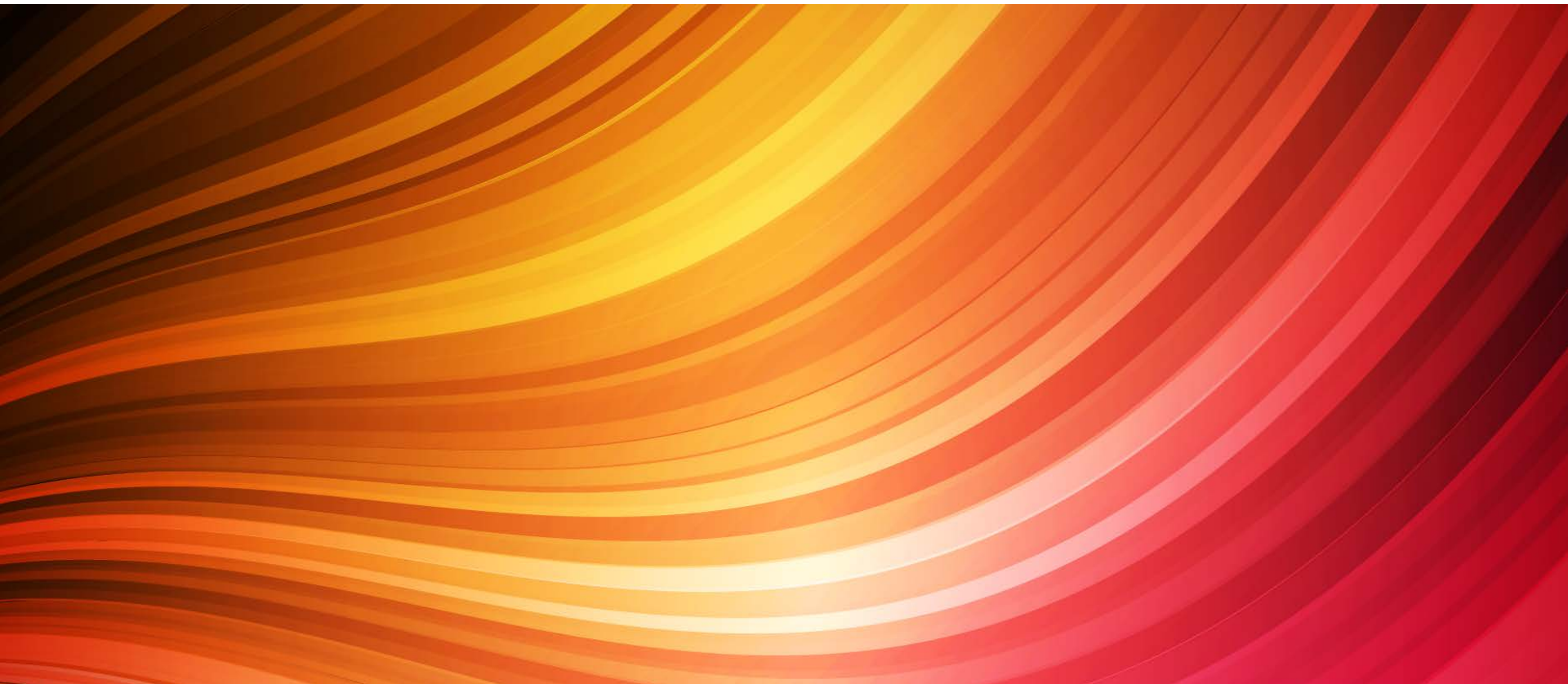


Migration between TarMK and MongoMK

- Backend choice mostly transparent to higher level code
- Migrating the full repository
 - use backup/restore
- Migrating selected subtrees
 - use content packages



Backwards compatibility



Significant changes

Functionality with significant changes in implementation and whose use in client code should be reviewed and, if needed, adapted

Query

- See next section

Observation

- Single-node observation mostly unchanged
- Commit boundaries, user info, etc. not available across cluster

Security

- Significant changes in performance and flexibility of access controls
- Backwards compatibility a priority, so few direct problems expected, but review for performance

Minor differences

Functional changes that in most cases require few or no changes in client code

Session refresh

- Sessions not always up to date with latest changes
- Auto-refresh feature avoids most compatibility issues

Identifiers

- Only referenceable nodes have UUIDs
- Other nodes have path identifiers

Versioning

- Different frozen identifiers
- Updated access control

Missing features

Functionality that is either completely missing or significantly less useful than before and thus require significant rewrite in affected client code

Workspaces

- Only a single workspace per repository
- Unneeded in normal AEM deployments

Same-name siblings

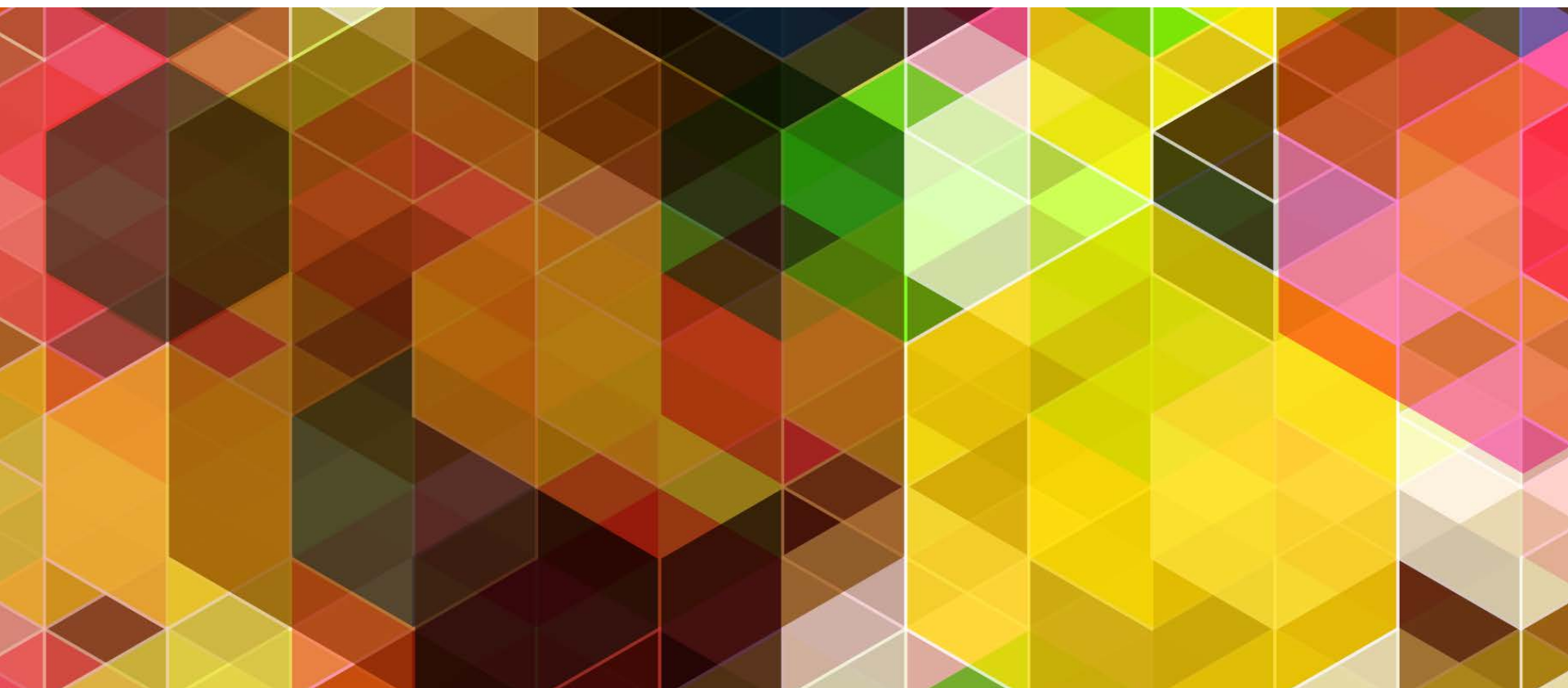
- Only very limited support (essentially read-only)
- Not very frequently used

Locking

- Only "soft locking" implemented
- Use as a guideline ("I'm working on this page") instead of as a strict synchronization tool



Custom search indexes



Search Engine

Features not covered here:

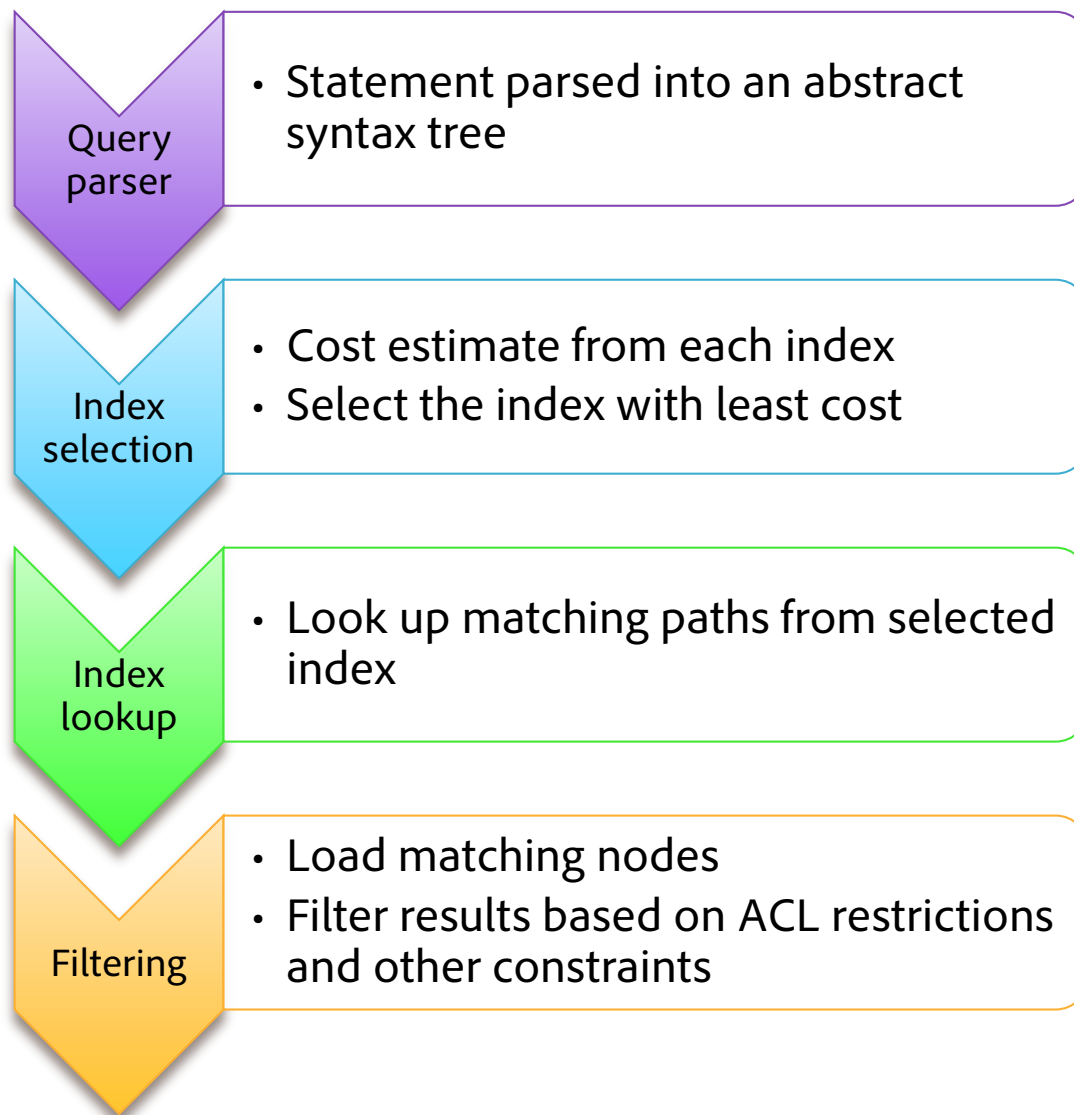
- joins
- ordering

Features not yet available:

- facets
- Aggregates

Trouble with a query?

- try the EXPLAIN feature!



Index definitions

Nodes under /oak:index

- Index data as hidden content or in an external location

Creating an index:

- add a definition node

Removing an index:

- remove the node

Re-indexing:

- set reindex=true

Update frequency:

- async="async"

Fallback:

- traversing index

The screenshot shows the CRXDE Lite web interface. The left sidebar displays a tree view of the repository structure, with the path `/oak:index/jcrLanguage` selected. The main content area shows the 'Properties' tab for this node. The properties table is as follows:

	Name	Type	Value	Protected	Mandatory
1	jcr:primaryType	Name	oak:QueryIndexDefinition	true	true
2	propertyNames	Name[]	jcr:language	false	false
3	reindex	Boolean	false	false	false
4	type	String	property	false	true

At the bottom of the interface, there is a form to add new properties with fields for Name, Type (set to String), Value, and checkboxes for Multi, Add, and Clear.

Property index

```
SELECT * FROM [mix:language]  
WHERE [jcr:language]=?
```

jcr:primaryType	Name	oak:QueryIndexDefinition
propertyNames	Name[]	jcr:language
reindex	Boolean	false
type	String	property

Constraints on multiple properties

```
SELECT * FROM [nt:base]  
WHERE foo=? AND bar=?
```

- one index on foo and another on bar
- engine automatically selects best index
- other constraints applied as extra filters
- future extension: multi-property index

Unique index

```
SELECT * FROM [rep:Authorizable]  
WHERE [rep:principalName]=?
```

declaringNodeTypes	Name[]	rep:Authorizable
jcr:primaryType	Name	oak:QueryIndexDefinition
propertyNames	Name[]	rep:principalName
reindex	Boolean	false
type	String	property
unique	Boolean	true

Ordered index

```
SELECT * FROM [nt:base]
WHERE [cq:lastModified] > ?
AND [cq:lastModified] < ?
ORDER BY [cq:lastModified]
```

<code>async</code>	String	<code>async</code>
<code>jcr:primaryType</code>	Name	<code>oak:QueryIndexDefinition</code>
<code>propertyNames</code>	Name[]	<code>cq:lastModified</code>
<code>reindex</code>	Boolean	<code>false</code>
<code>type</code>	String	<code>ordered</code>

Lucene / Solr index

```
SELECT * FROM [nt:base]
WHERE CONTAINS(*, ?)
```

Lucene index files stored
as hidden content inside
the repository!

Solr index in an external
search server.

async	String	async
excludePropertyNames	String[]	analyticsProvider, analyticsSnip
includePropertyTypes	String[]	String, Binary
jcr:primaryType	Name	oak:QueryIndexDefinition
reindex	Boolean	false
type	String	lucene



Adobe