# Introduction to JBoss DNA

*September 2009*

# JBoss DNA

- Open-source project
- Content repository
- Supports the JCR API
- Support a REST API
- Federates and unifies content
- Automation to make content useful

# Our open source community is everything

It’s where we make noise

It’s where users find us

It’s where users find the software

It’s where we support our users

It’s where we decide where we’re going

It’s where we collaborate and contribute

It’s where we do everything!

**http://jboss.org/dna**

# The 411 on JBoss DNA

- Project site - http://jboss.org/dna/
- Blog - http://jbossdna.blogspot.com/
- JIRA issue tracker - http://jira.jboss.org/jira/browse/DNA
- IRC chat - irc.freenode.net#jbossdna
- Forums - http://jboss.org/index.html?module=bb&op=viewforum&f=272
- Mailing lists - http://jboss.org/dna/lists
- Documentation - http://jboss.org/dna/docs

# Why does JBoss do this?

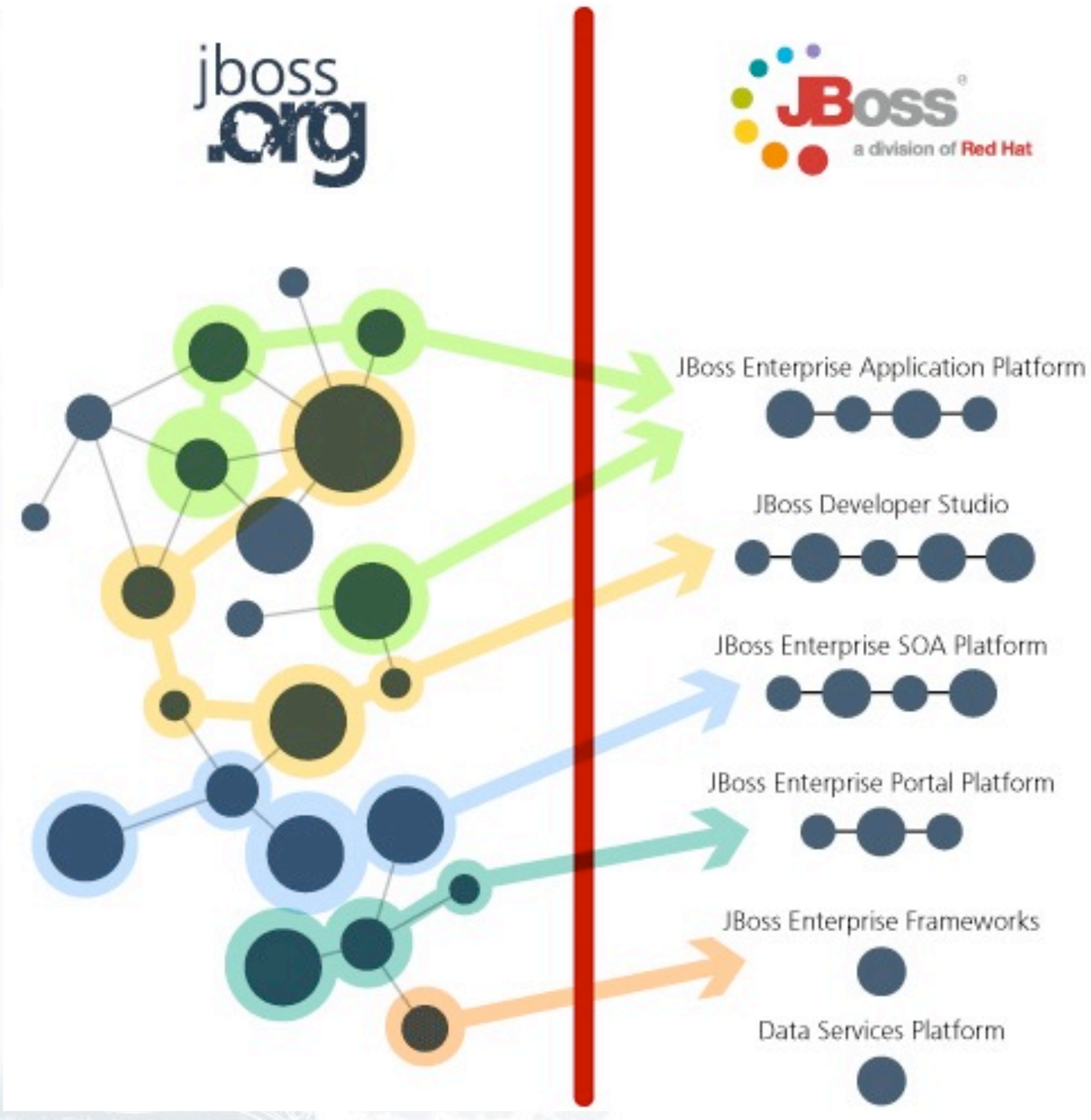
*community support*

*80+ projects*

*80+ schedules*

*release early, release often*

*where innovation happens*




*subscription-based support*

*integrated & certified distributions*

*extensive QA*

*updates & patches*

*stability*

# “Content management”

- Umbrella term for industry-specific classes
  - Digital Asset Management
  - Web Content Management
  - Digital Records Management
- Create, edit, manage, search and public digital media and electronic text
- Often includes
  - workflow
  - version management
  - content capture

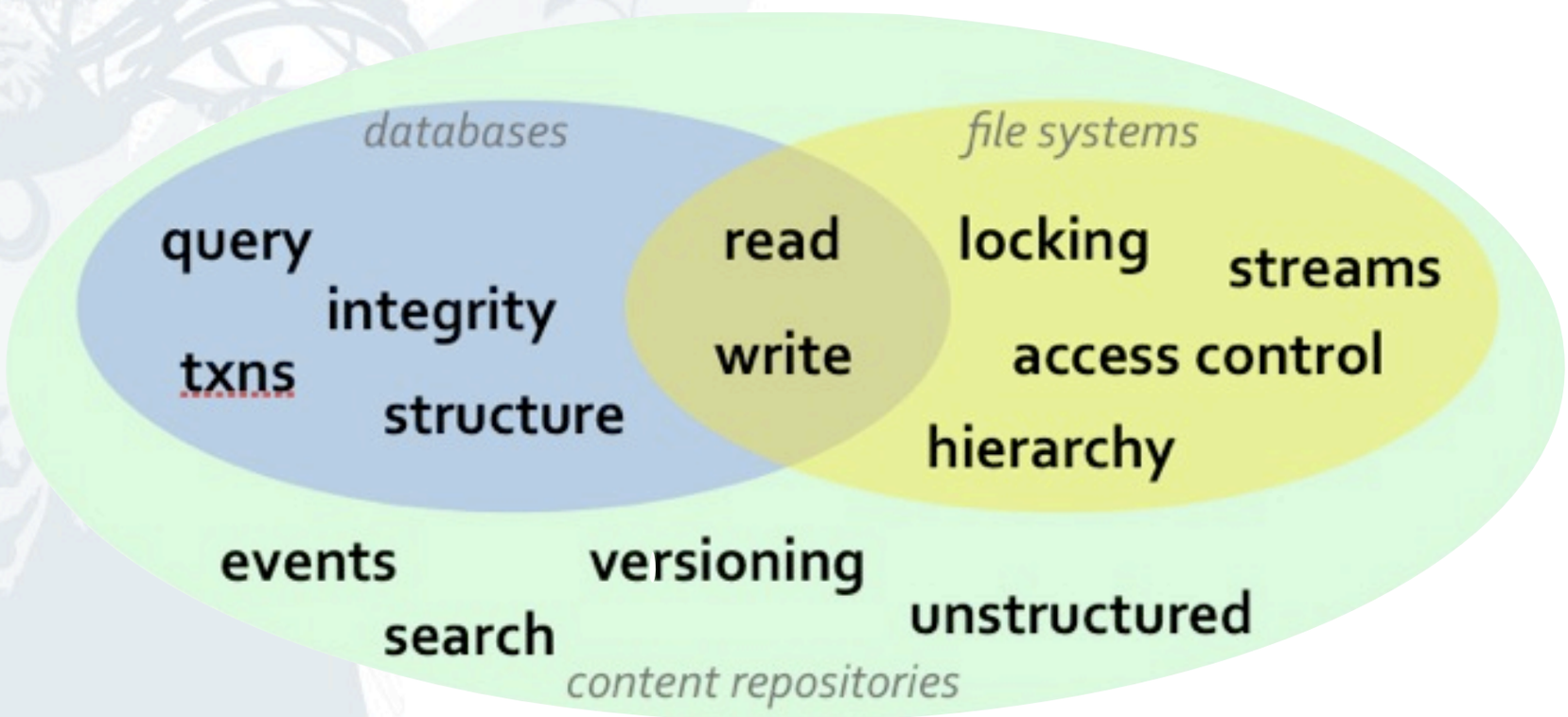
# “Content repository”

- Where CMS system stores content
- Hierarchical graph-based storage
  - flexible schema
  - handles wide variety of content
  - adapts to change
- Versioning, events, access control, transactions
- Search and often query

# Java Content Repository (JCR)

JSR-170 and JSR-283

- Standard programmatic API: `javax.jcr`
- Abstracts where/how the content is stored
- Best features from databases and file systems



# Who uses JCR?

JBoss a division of Red Hat or jboss.org

magnolia

CRX

mulegalaxy

Alfresco

eXo platform

InfoQ Enterprise Software Development Community

Freecaster.tv

Xythos software, inc.

SOPHORA

ORACLE

Nuxeo 5

The Apache Software Foundation http://www.apache.org/

OpenPortal

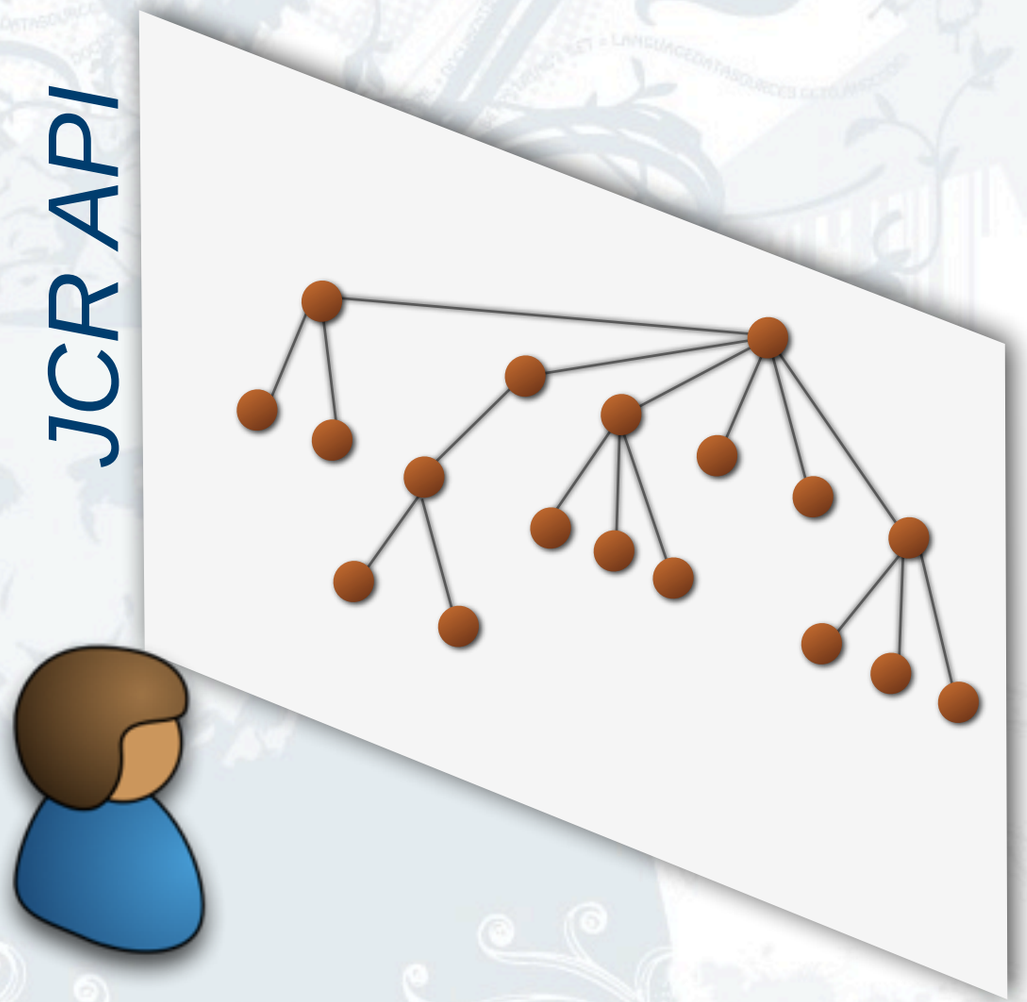
LIFERAY Enterprise. Open Source. For Life.

IBM

Rhythmyx Content-Driven Applications

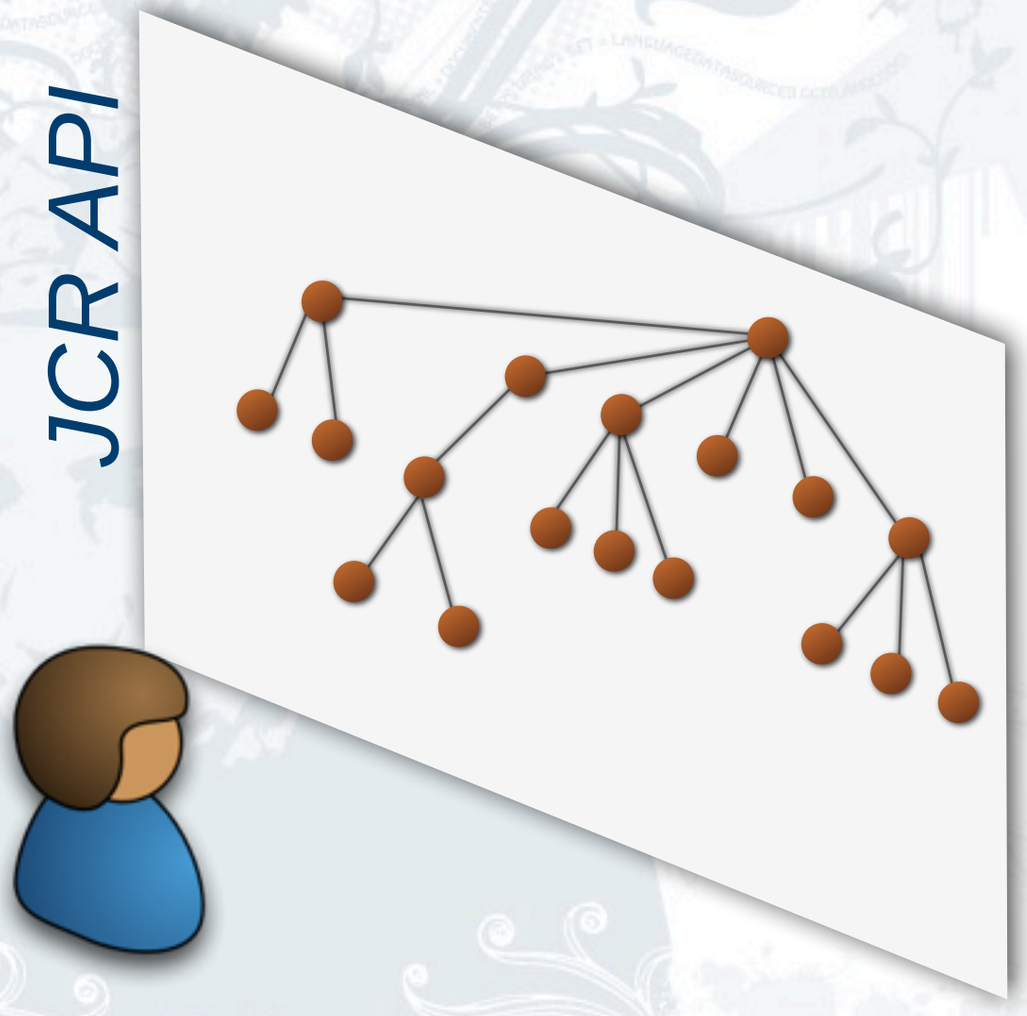
***... and many more***

# JCR repositories




- Flexible schema
  - data heterogeneity
  - your data WILL change
- Hierarchical content
- Versioning
- Events
- Transactions
- Search and query

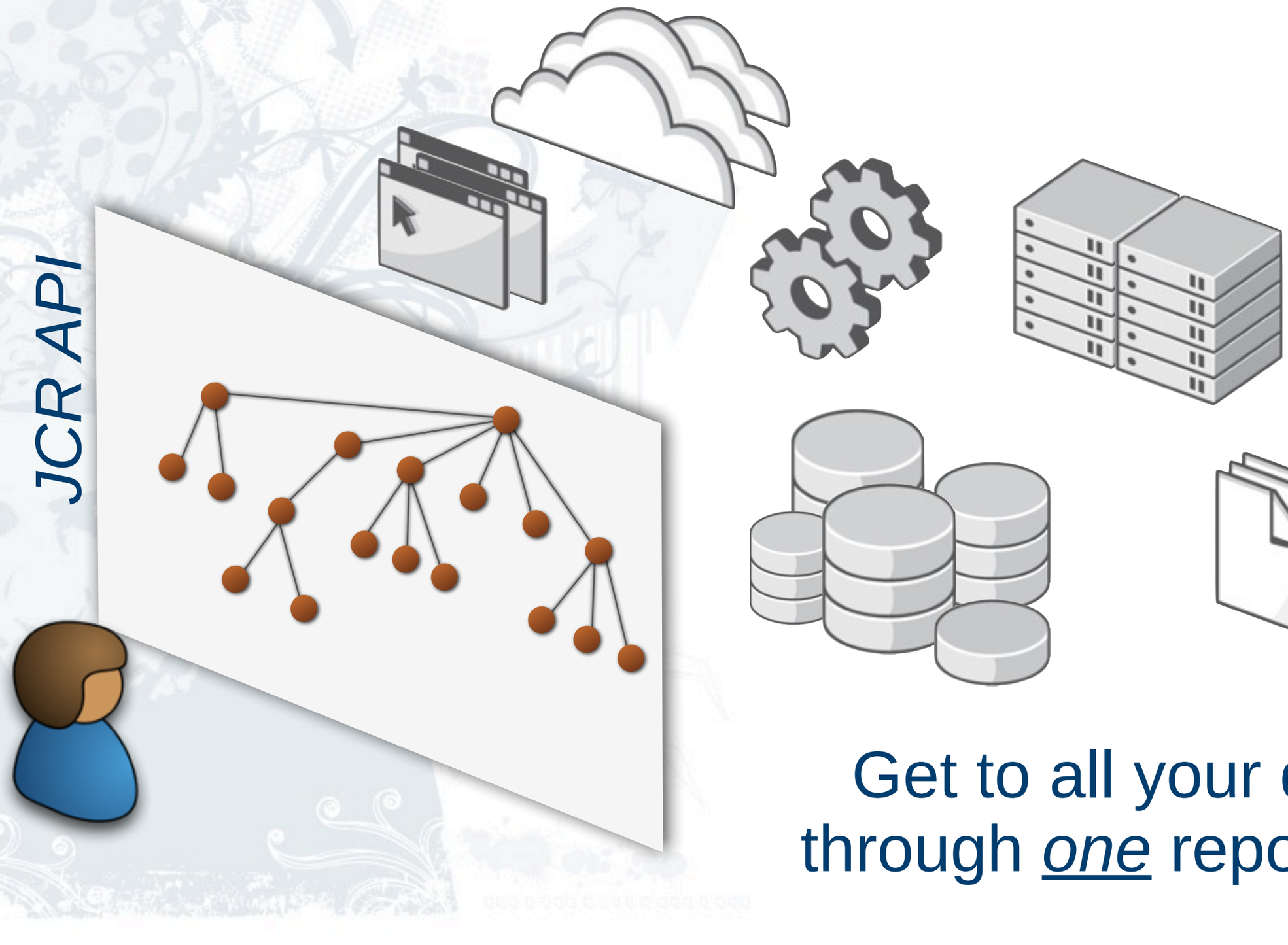
# Other JCR repositories …




Store all your data
in *one* silo*

* though they sometimes offer
different kinds of silos

# JBoss DNA




Get to all your data
through *one* repository

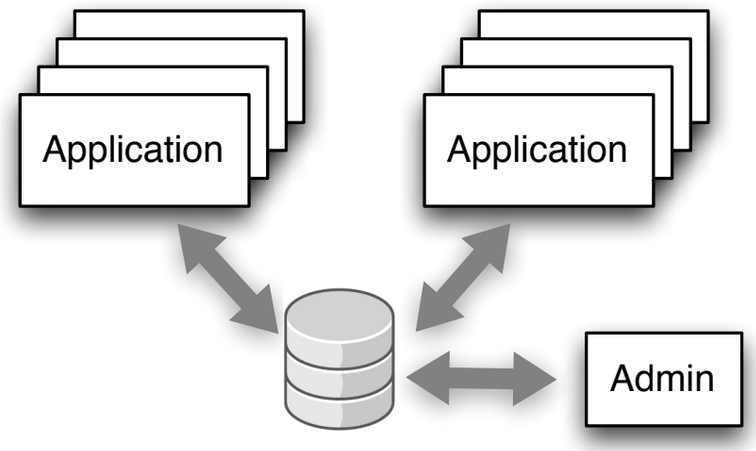
# A few ways to use JBoss DNA

# “Schema-less” Data

- Let the data drive the structure
  - Data schema can easily evolve over time
  - Data can have variations
- Additions to structure don’t (usually) affect older software versions
- Can still have types (called “mixins”)
- Analogous to dynamic programming languages
- Free search and query

# Application configuration

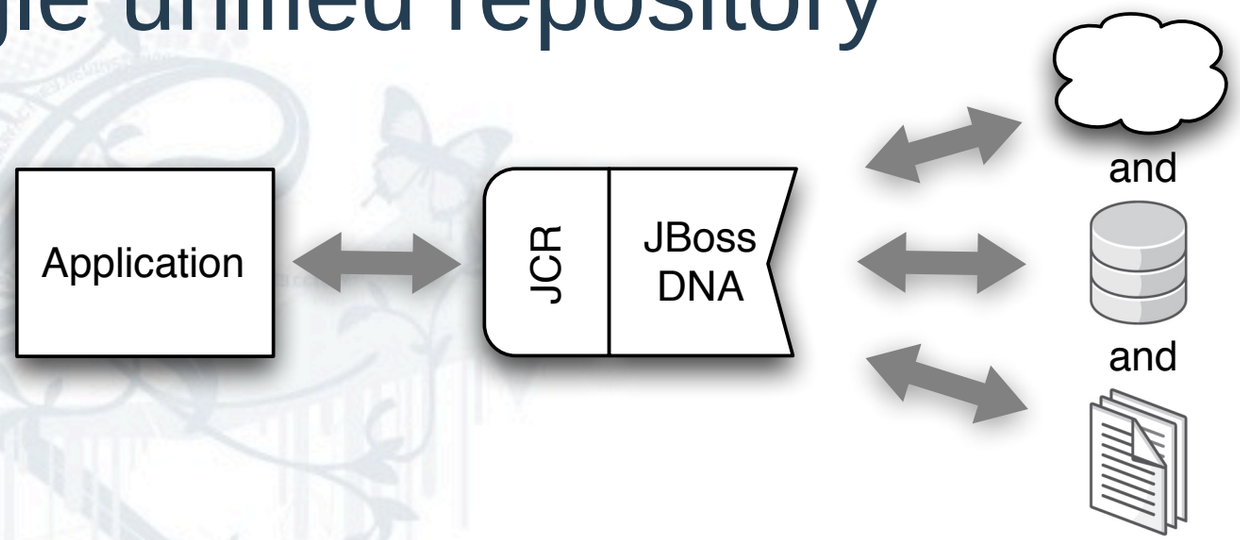
- Application doesn’t care where it’s stored
  - local files (e.g., XML) or shared central repository


- Can leverage versioning
- Events notify when specific areas of configuration changes
- Allows changing configuration structure

# Federation

- Access existing content as if it existed in a single unified repository

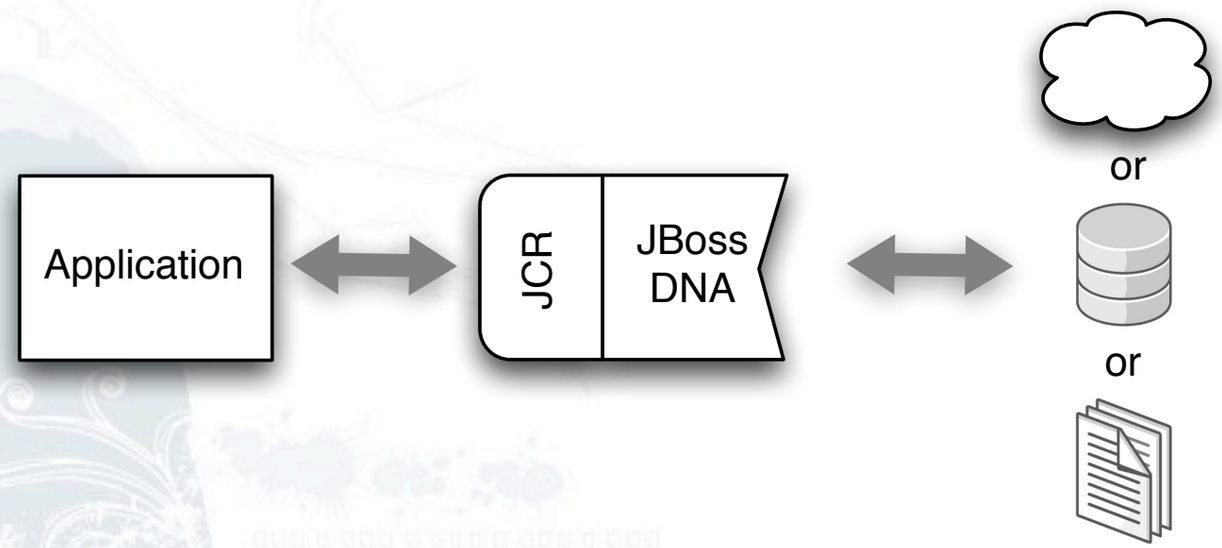



- Read and update *
- Receive events when data changes
- Search and query
- Can locally cache accessed content

* depends on capability of connectors and how they're configured

# Use the cloud

- Access data regardless of where it is
  - databases, file systems (Amazon EC2), data grids (Infinispan), clouds (Amazon S3, Google Data)
- Use standard JCR API
- Allows redeployment in different environments
  - application servers, services, clouds



# Project status

- Released 0.6 on Sept 13 2009
  - Almost full JCR L1 & L2 compliance
  - Connectors to DBMS, in-memory, Infinispan, SVN, file system, JBoss Cache, federation
  - Sequencers for CND, Java source, ZIP, XML, Microsoft Office files (doc, ppt, xls), images, MP3
  - REST servlet

# Project roadmap

- 1.0 release (late Oct, early Nov)
  - Full JCR L1 & L2 compliance
  - Includes SQL & XPath query + full-text search
  - Ability to push-down queries/searches to connectors
  - Eclipse “publishing” plug-in uses REST server
- 1.1 and beyond about 4 weeks
  - JCR optional features
  - other features, fixes, etc.

# For more information

- www.jboss.org/dna
- Downloads (version 0.3)
  - Binary, source, documentation, examples
- JBoss Maven 2 Repository
  - “org.jboss.dna” group ID (several artifacts)
- Documentation
  - Getting Started describes the design, the different components, and how to use them with a trivial example application
  - Reference Guide describes how JBoss DNA works internally from a developer perspective
- Blogs: jbossdna.blogspot.com

# Questions?

# Thanks!

# Configuration

- Fluent API
- Step 1: Load from file, a configuration repository, or create programmatically
- Step 2: Create the engine
- Step 3: Get JCR repositories & use JCR API
- Step 4: Shut down engine

# Step 1: Loading configuration

- Load from file:

```
JcrConfiguration config = new JcrConfiguration();
configuration.loadFrom(file);
```

- or load from repository:

```
RepositorySource configSource = ...
JcrConfiguration config = new JcrConfiguration();
configuration.loadFrom(configSource);
```

- or ...

# Step 1: Loading configuration

- ... or define programmatically:

```
JcrConfiguration config = new JcrConfiguration();
config.repositorySource("source A")
      .usingClass(InMemoryRepositorySource.class)
      .setDescription("The repository for our content")
      .setProperty("defaultWorkspaceName", workspaceName);
config.repository("repository A")
      .addNodeTypes("myCustomNodeTypes.cnd")
      .setSource("source 1")
      .registerNamespace("acme",
                 "http://www.example.com/acme")
      .setOption(Option.JAAS_LOGIN_CONFIG_NAME,
                 "dna-jcr");
...
```

# Step 1: Loading configuration

- ... or define programmatically (cont'd):

```
JcrConfiguration config = ...
config.sequencer("Image Sequencer")
      .usingClass("com.example.MySequencer")
      .loadedFromClasspath()
      .setDescription("Sequences image files")
      .sequencingFrom("//(*.(jpg|jpeg|gif)[*])/jcr:content[@jcr:data]")
      .andOutputtingTo("/images/$1");
...
config.mimeTypeDetector("Extension Detector")
      .usingClass(“com.example.MyMimeTypeDetector”)
      .loadedFromClasspath();
```

# Step 2: Create the engine

- Create and start the engine

```
JcrConfiguration config = ...
JcrEngine engine = config.build();
engine.start();
```

# Step 3: Use the repositories

- Get the JCR repositories by name

```
javax.jcr.Repository repository =
          engine.getRepository("Name of repository");
```

- Everything else is standard JCR API

```
javax.jcr.Session session = repository.login(...);
...
```

# Step 4: Shut down

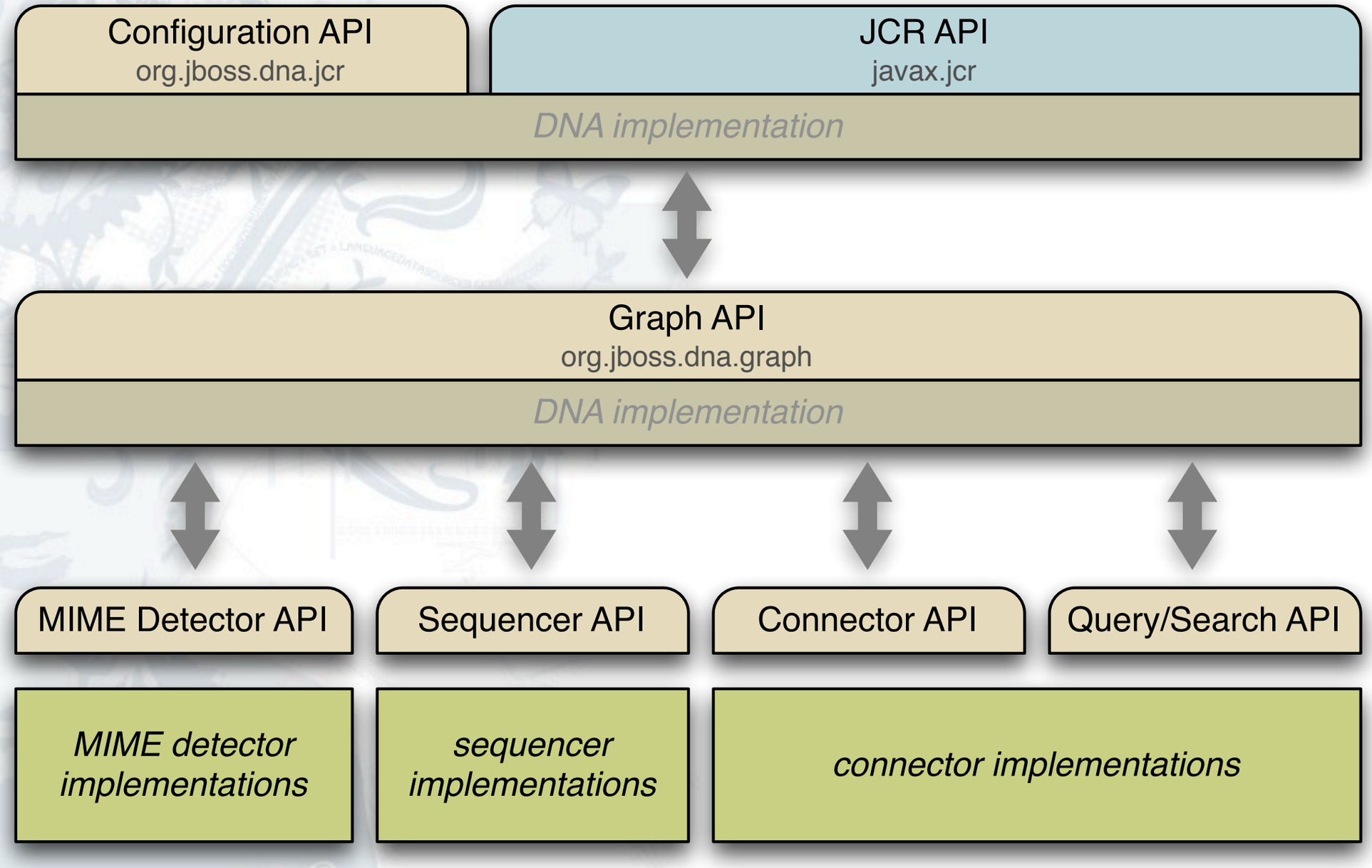
- When finished, shutdown the engine

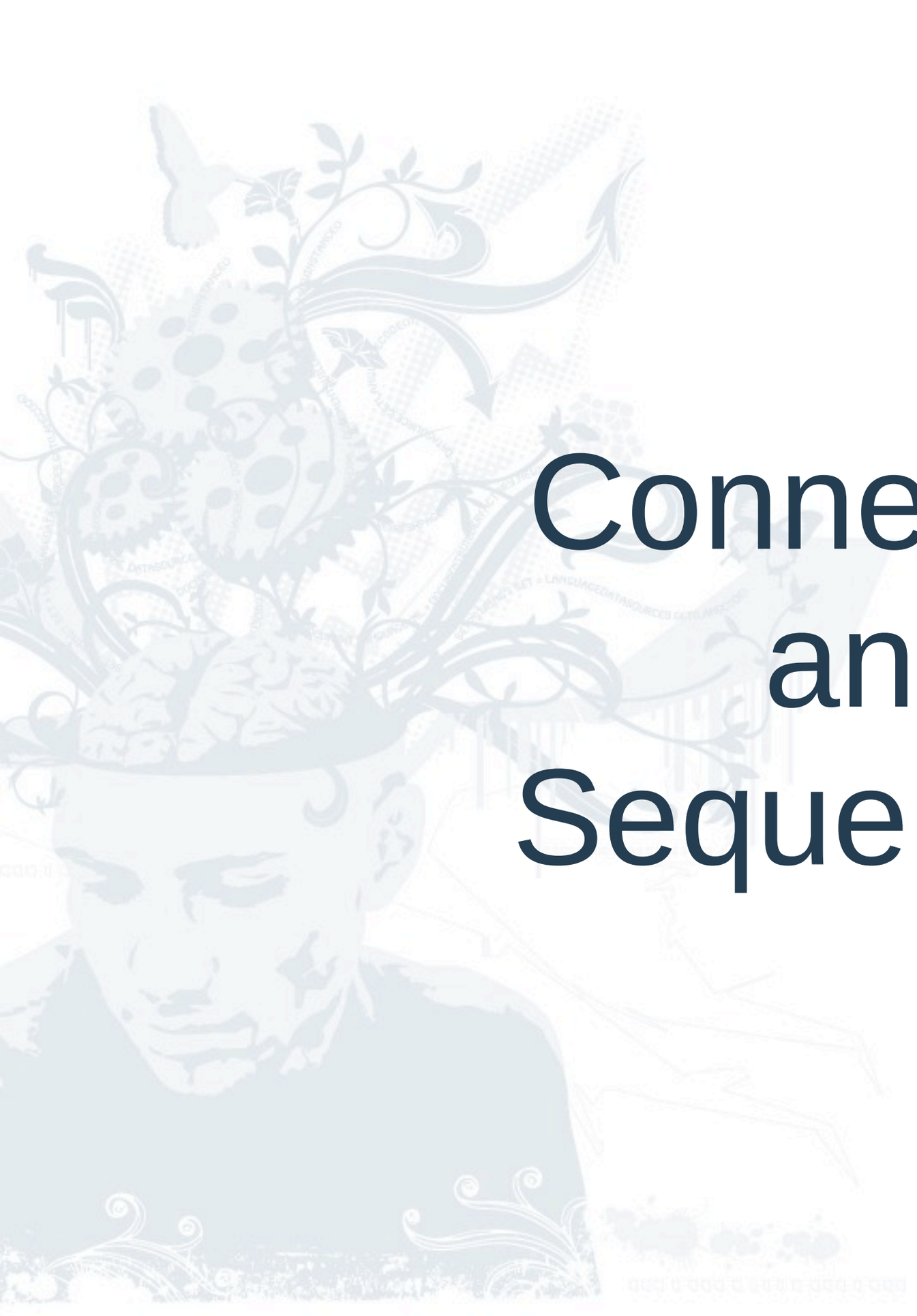
```
engine.shutdown();
```

- Optionally await until everything completes naturally

```
engine.awaitTermination(3,TimeUnit.SECONDS);
```

# JBoss DNA architecture

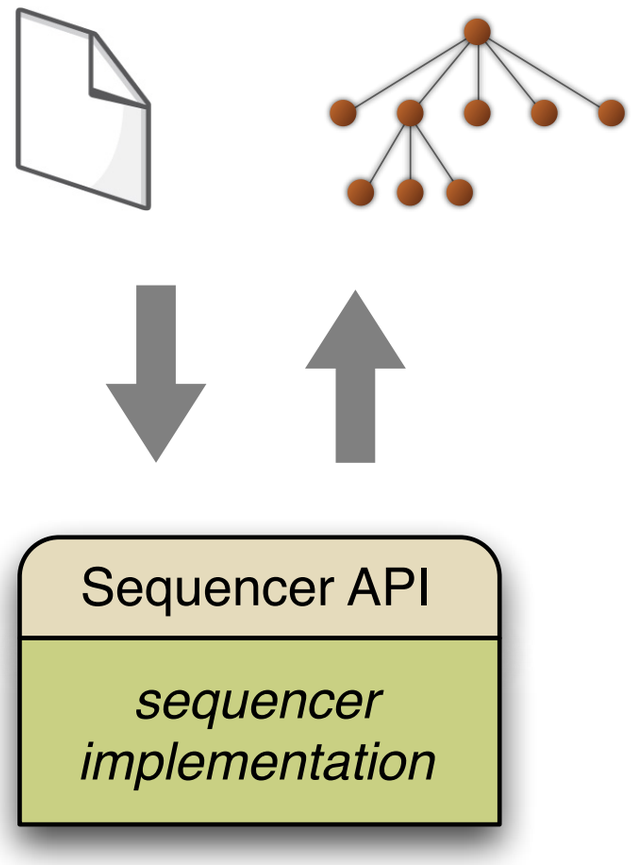




# Connectors and Sequencers

# Sequencers

- Content is changed
- JBoss DNA determines which (if any) sequencers apply
- JBoss DNA notifies appropriate sequencer(s) with content to be sequenced
- Sequencer processes content and generate graph
- JBoss DNA places generated content into repository



# JBoss DNA sequencers
(current)

- CND
- XML
- Microsoft Office files
- Java source
- Images
- ZIP files
- MP3 audio files


32

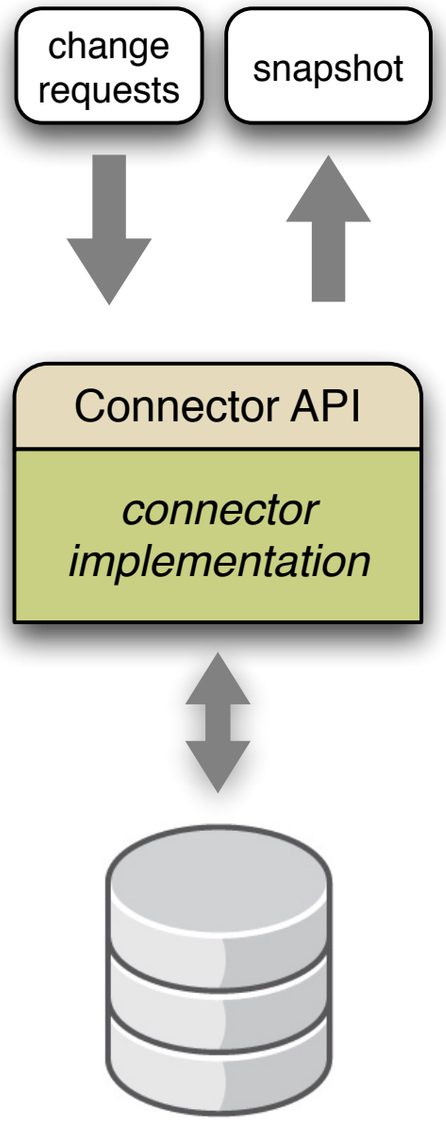
# JBoss DNA sequencers

(in-work or planned)

- DDL files
- JBoss ESB messages
- JBoss PDL files
- Drools rules
- MetaMatrix models
- Maven POM
- Maven test results
- Hibernate mappings
- Java class files
- Microcontainer metadata
- App Server configurations
- ANTLR grammars
- OSGI manifests

# Connectors

- Liaison between JBoss DNA and external systems
- Receive read/write requests
- Translate requests into system's native protocol
- Perform requested operation and (if requested) return snapshot of the requested state



# JBoss DNA connectors

(current)

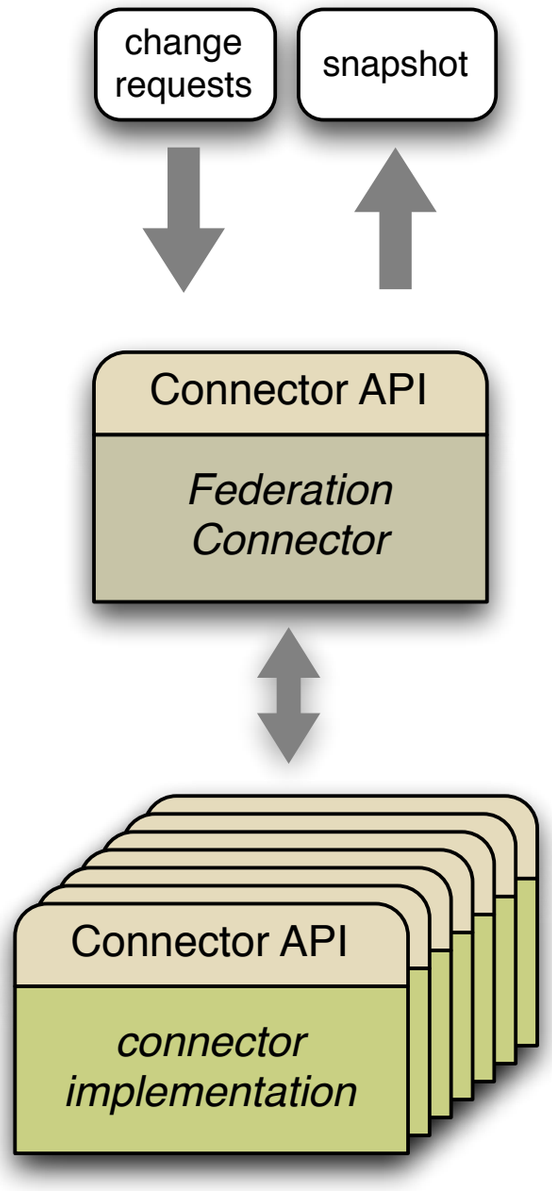
- **In-memory**: simple transient repository
- **Federation**: projects content from 1+ sources into a single repository
- **Infinispan**: persists content in Infinispan data grid
- **Relational store**: persists content in database using JPA/Hibernate
- **JBoss Cache**: for local (clustered) cache of content
- **File System**: projects files/folders as content
- **SVN**: access the files under version control



35

# Federation Connector

- Appears to be a standard connector
- Projects content from multiple other connectors into a single unified graph
- Supports read and write
- Optimized for certain common configurations (e.g., mirror, mirror-branch, offset, etc.)

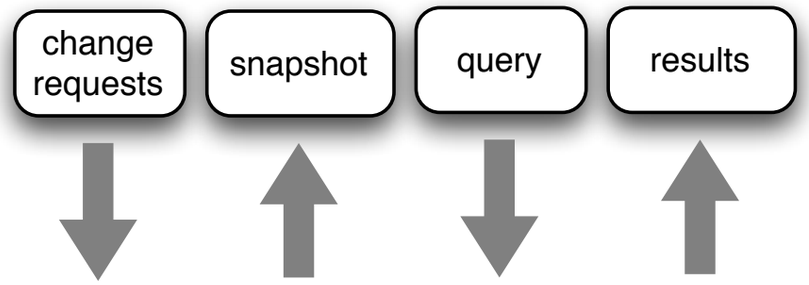


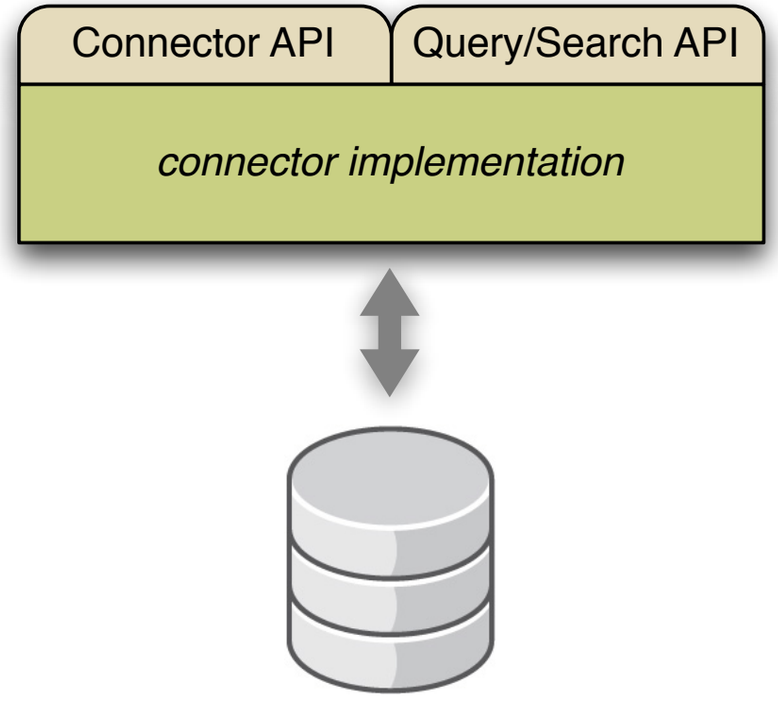
# Connectors with search/query

*(in trunk)*

- Optionally support the Query/Search API
- Receive queries, which include full query capabilities
- Build and return result set

*Connectors can do this in any way*



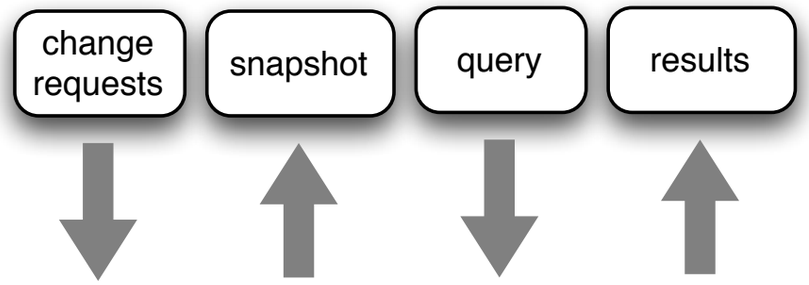



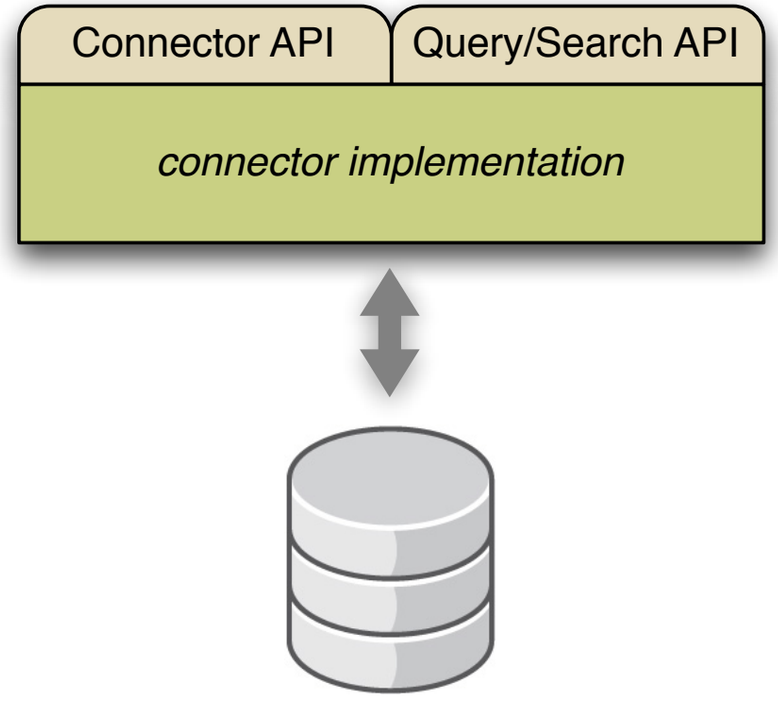
# Connectors with search/query

*(in trunk)*

- Optionally support the Query/Search API
- Receive queries, which include full query capabilities
- Build and return result set

*Connectors can do this in any way*

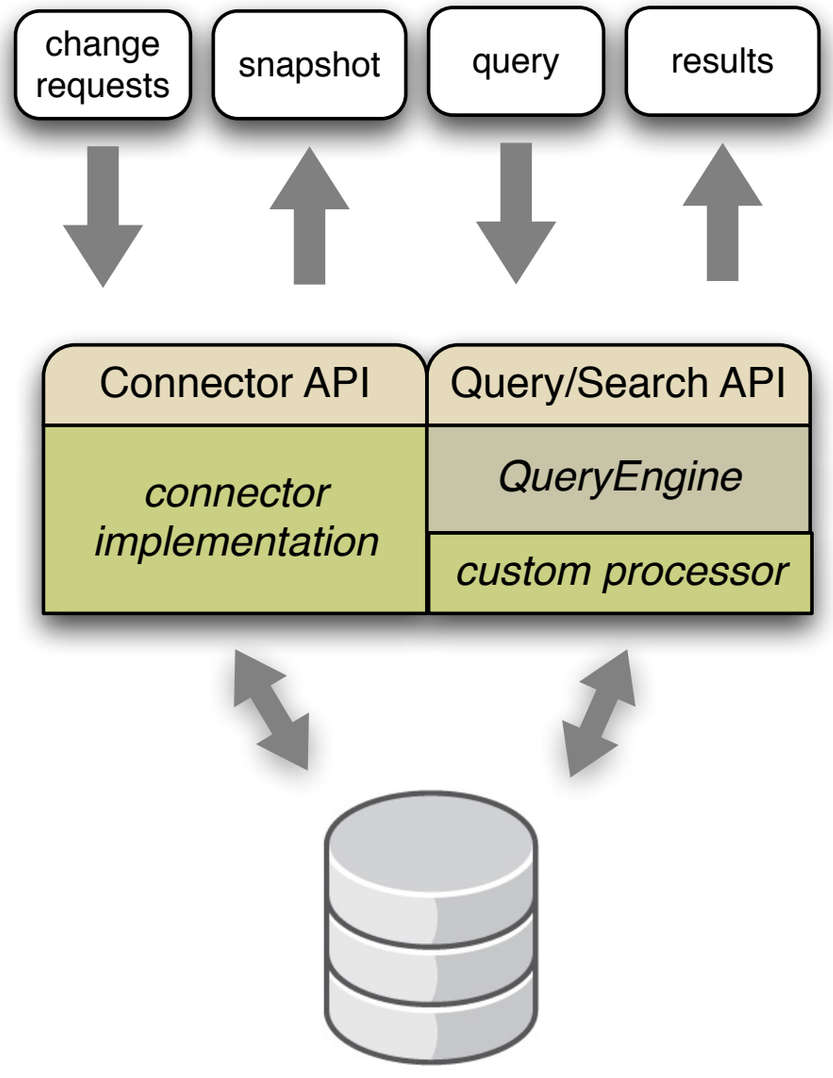


# Connectors with search/query

*(in trunk)*

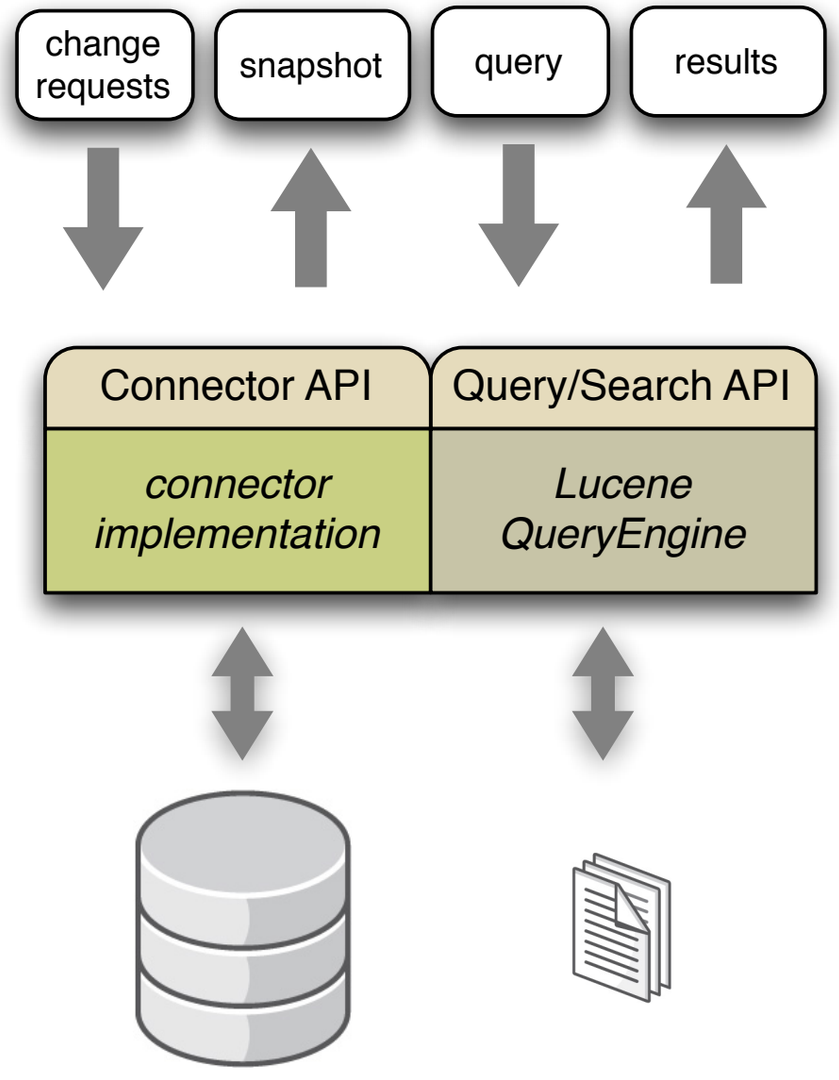
- JBoss DNA provides a default QueryEngine implementation that supports all query capabilities
- Connector provides a custom QueryProcessor that does simple SELECTs



# Connectors with search/query

*(in trunk)*

- JBoss DNA provides a Lucene-based QueryEngine implementation that supports all query capabilities
- Connector must simply manage this instance and call its methods to generate/ update indexes



# JBoss DNA connectors

## (in-work or planned)

- **JCR**: access the content from another JCR repository
- **JDBC Metadata**: access the database metadata (catalogs, schemas, tables, columns, keys, etc.)
- **Git**: access the files under version control
- **Maven**: accesses the artifacts in a Maven 2 repository
- **UDDI**: accesses the registry content
- **LDAP**: accesses the directory content
- **Informatica**: access the metadata repository
- **Cognos**: access the metadata from Cognos
- ... and others



41