
codecentric
agile software factory
SPRING XD
Tackling Big Data Complexity

Dennis Schulte

codecentric
Düsseldorf

@denschu
www.github.com/denschu
blog.codecentric.de/author/dsc

dennis.schulte@codecentric.de
www.codecentric.de

Tobias Flohre

codecentric

Düsseldorf

@TobiasFlohre

www.github.com/tobiasflohre

blog.codecentric.de/en/author/tobias.flohre

tobias.flohre@codecentric.de

www.codecentric.de

HERAUSFORDERUNGEN

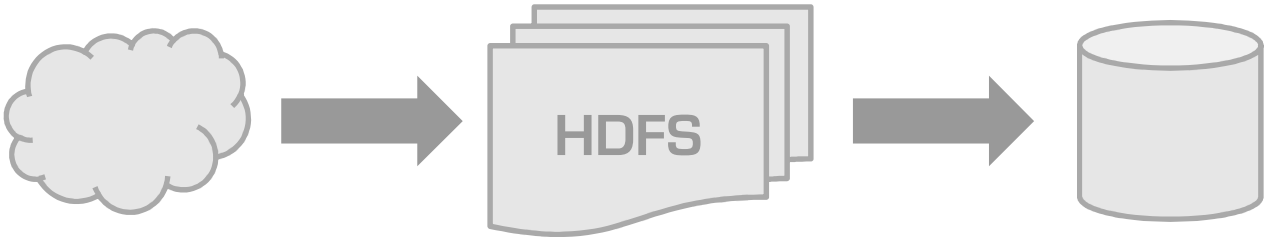
Datenmenge

Stream Processing und Real Time Analytics

Heterogene Datenquellen und Datenformate

WAS FOLGT DARAUS?

Datenintegration wird anspruchsvoller



Datenquellen

Transformation

Real-Time-Auswertungen

Skalierung

VORHANG AUF FÜR SPRING XD

Spring XD ist ein
einheitliches,
verteiltes und
erweiterbares
System für
Datenaufnahme,
Real-Time-Analytics,
Batch-Verarbeitung und
Datenexport

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

- Laufzeitumgebung
  - Kein Framework!
- Deployment spezieller Komponenten

VORHANG AUF FÜR SPRING XD

Spring XD ist ein

einheitliches,

verteiltes und

erweiterbares

System für

Datenaufnahme,

Real-Time-Analytics,

Batch-Verarbeitung und

Datenexport

- Laufzeitumgebung
  - Kein Framework!
- Deployment spezieller Komponenten
- Einheitliche DSL
- Zugriff auf unterschiedlichste Technologien
- One-Stop-Shop für Big-Data-Anwendungen

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

- Verteilte Laufzeitumgebung
  - Skalierbar!

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

- Verteilte Laufzeitumgebung
  - Skalierbar!
- Viele vorgefertigte Komponenten
- Komponenten können selbst geschrieben werden
- Motto: Einfache Dinge sollten einfach sein, komplizierte Dinge möglich

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

- Datenströme (Streams)
  - Beliebige Technologien
  - Einfache DSL
  - Spring Integration

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

- Datenströme (Streams)
  - Beliebige Technologien
  - Einfache DSL
  - Spring Integration
- Abhören von Datenströmen

VORHANG AUF FÜR SPRING XD

Spring XD ist ein einheitliches, verteiltes und erweiterbares System für Datenaufnahme, Real-Time-Analytics, Batch-Verarbeitung und Datenexport

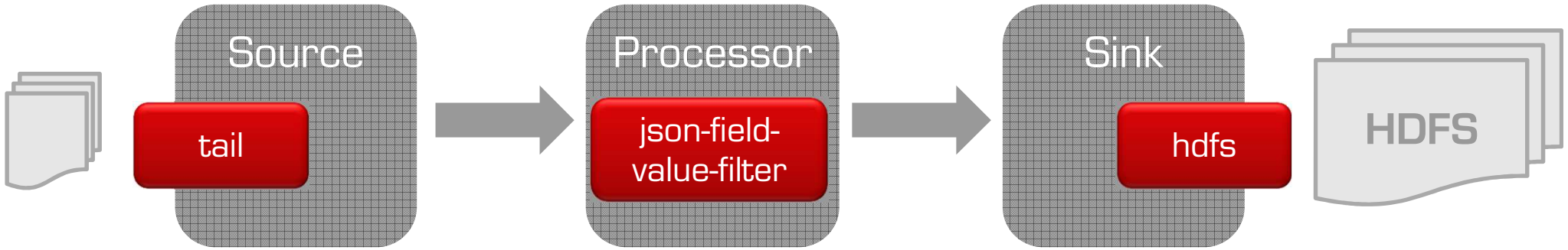
- Datenströme (Streams)
  - Beliebige Quellen und Ziele
  - Einfache DSL
  - Spring Integration
- Abhören von Datenströmen
- Jobs für Massenverarbeitung
  - Spring Batch
  - Spring for Hadoop


IO EXECUTION
Spring XD
Boot
Grails
IO FOUNDATION
Integration
Batch
Big Data
Web
Relational
DATA
Non-Relational
CORE
Framework
Security
Groovy
Reactor

AGENDA

- Streams
- Architektur
- Taps (Real-Time-Analytics)
- Jobs
- Erweiterbarkeit

STREAMS

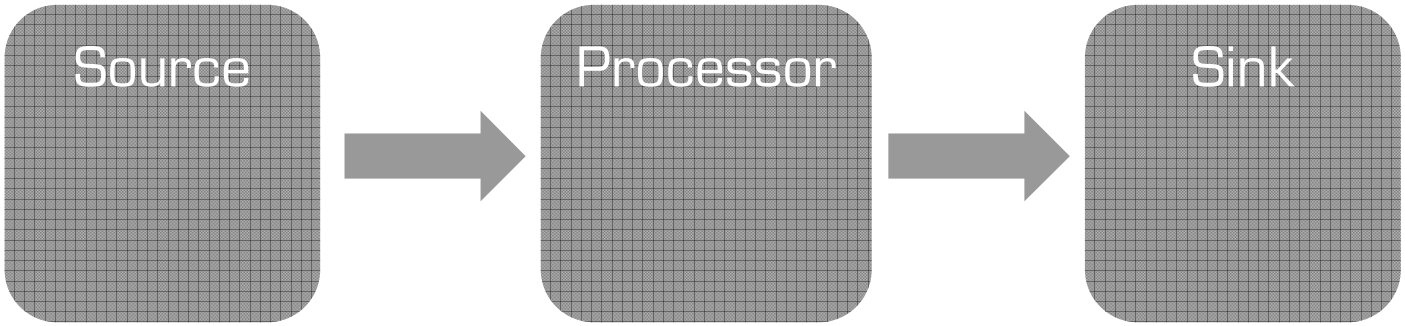


{"category":"INFO",
"url":"http://server.local:8080/api",
"duration":"200",
"timestamp":"1354697760477",
"http_code":"200",
"message":"This is a logentry."}

- Spring Integration – Module
- Einfache DSL

```
tail –name=/tmp/logfile.json
        | json-field-value-filter –fieldName=category –fieldValue=ERROR
                | hdfs
```

STREAMS

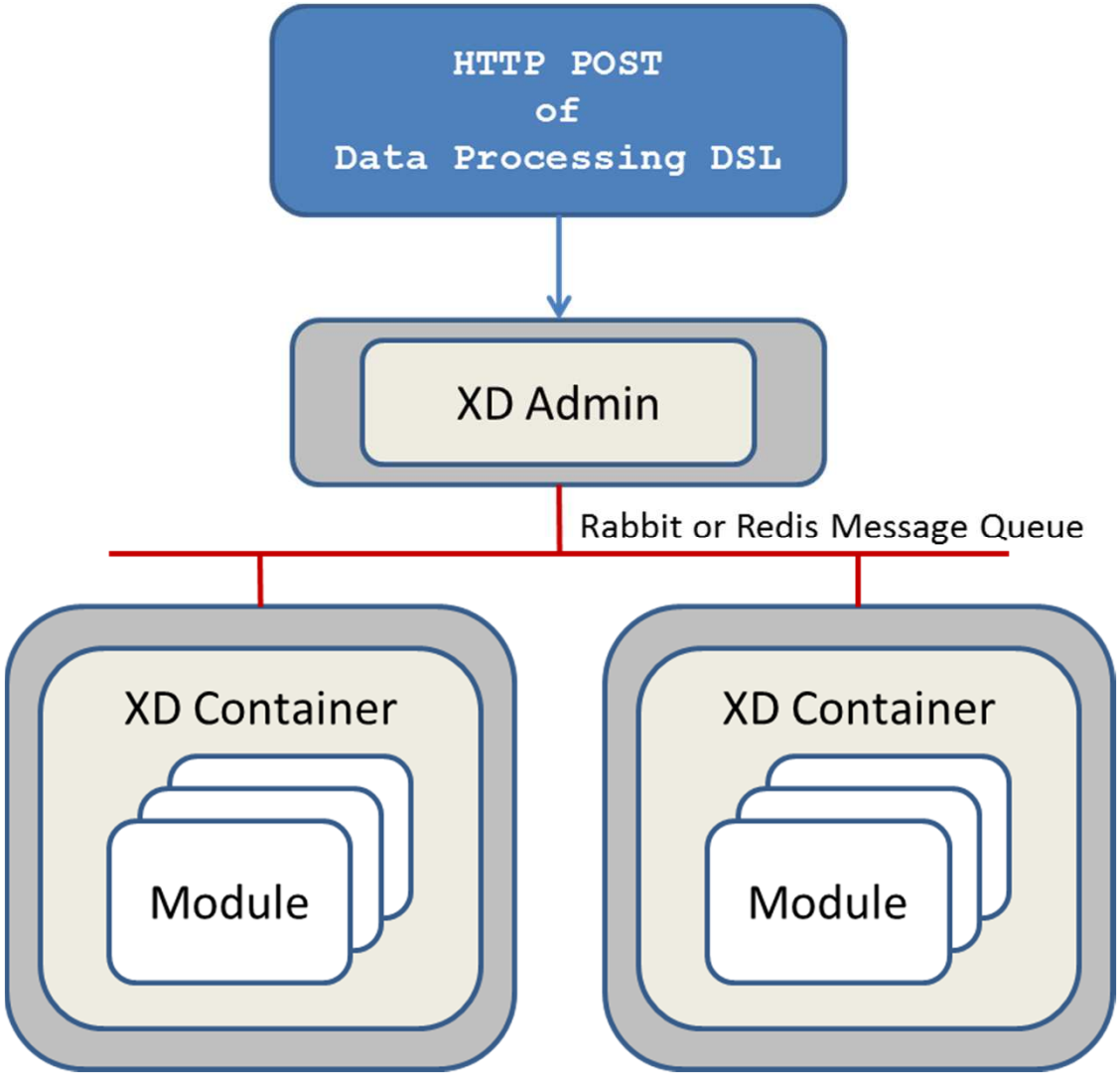


HTTP
JMS
TCP
RabbitMQ
Tail
Mail
Twitter
MQTT

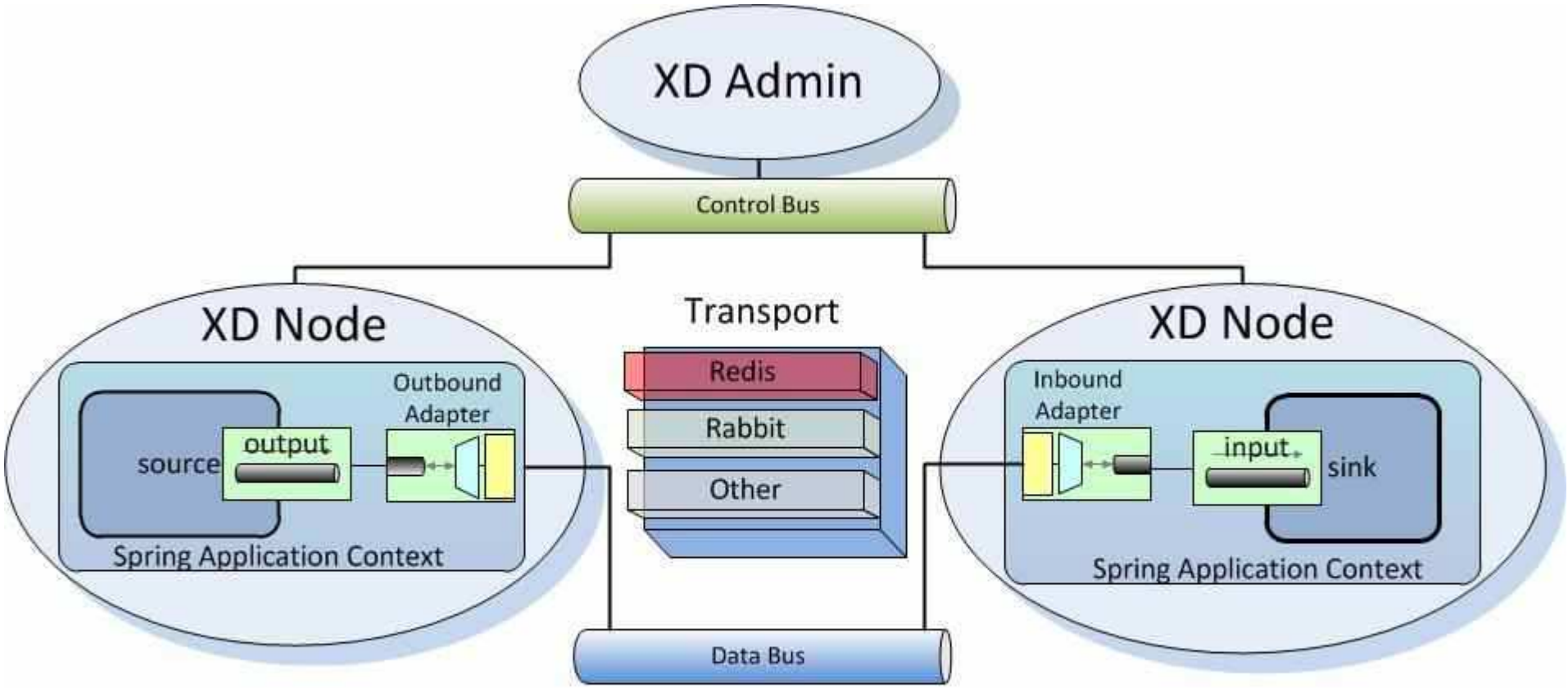
Filter
Splitter
Aggregator
Transform
JSON Field Extractor
JSON Field Value Filter

Log
File Sink
Hadoop (HDFS)
JDBC
TCP
Mail
Splunk
MQTT

ARCHITEKTUR



ARCHITEKTUR



<source> | <sink>

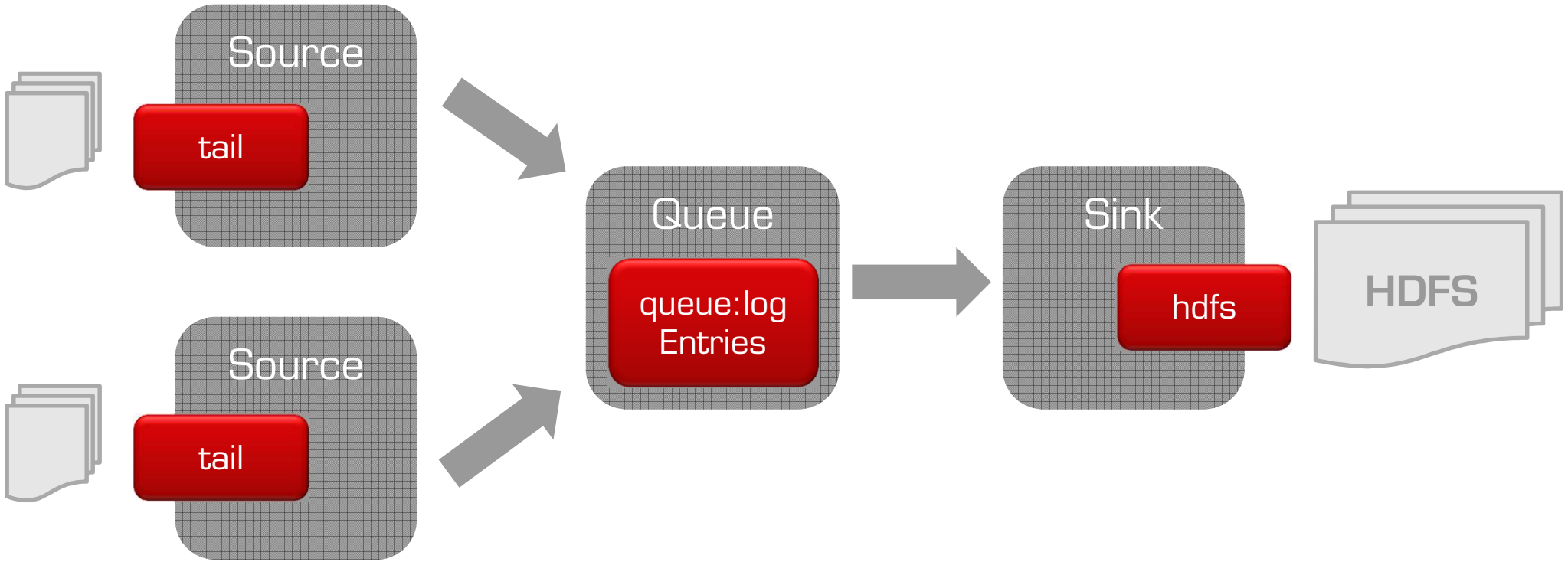
ARCHITEKTUR

XD-Shell

```
xd:> stream create –name beispiel –definition "tail –name=/tmp/logfile.json | hdfs"
xd:> stream destroy –name beispiel
xd:> stream undeploy –name beispiel
xd:> stream deploy –name beispiel
```

DEMO

STREAMS

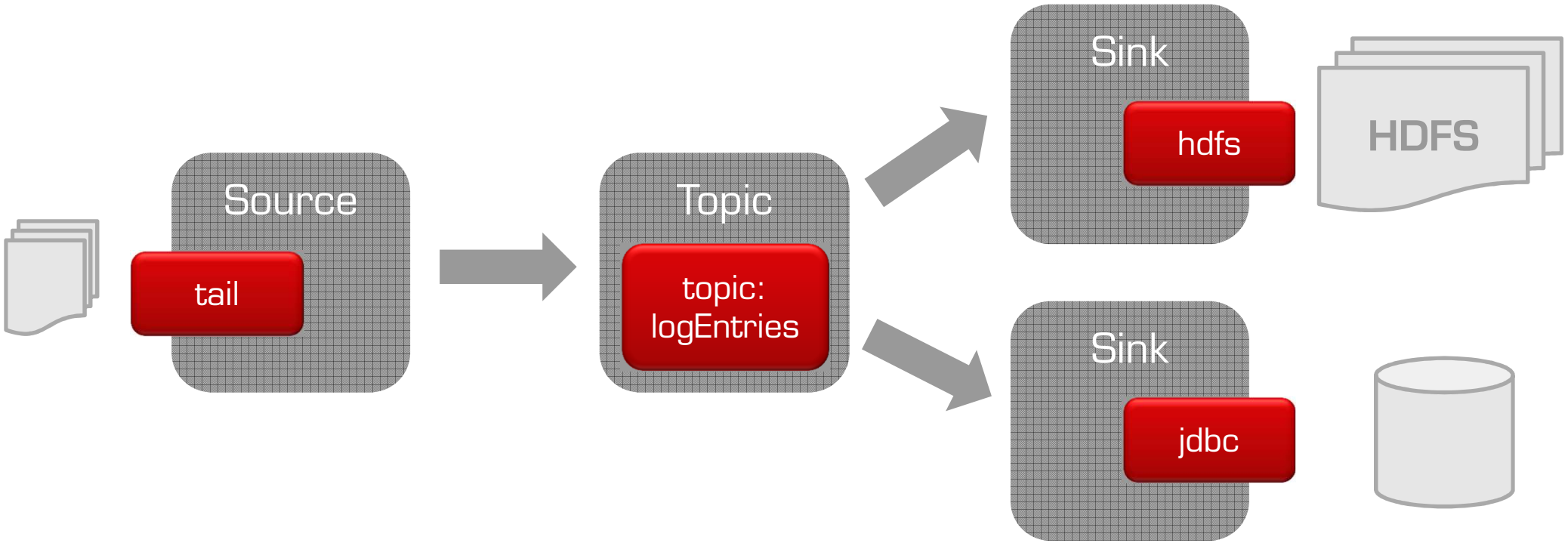


tail –name=/tmp/logfile1.json > queue:logEntries

tail –name=/tmp/logfile2.json > queue:logEntries

queue:logEntries > hdfs

STREAMS



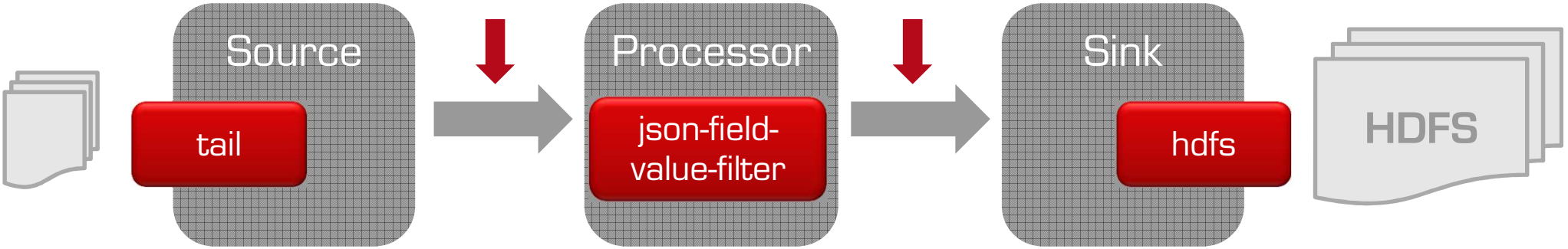
tail –name=/tmp/logfile.json > topic:logEntries

topic:logEntries > jdbc

topic:logEntries > hdfs

TAPS (REAL-TIME-ANALYTICS)

xd:> stream create –name beispiel –definition "tail (…) | json-field-value-filter (…) | hdfs"



tap:stream:beispiel > counter –name=requestcount

tap:stream:beispiel > json-field-extractor –fieldName=duration | richgauge –name=duration

Durchschnitt / Maximum / Minimum / Anzahl

tap:stream:beispiel.json-field-value-filter > counter –name=requestcount

TAPS (REAL-TIME-ANALYTICS)

Analytics-Module

Counter

Field Value Counter

Aggregate Counter

Gauge

Rich Gauge

Speicherung der Werte in Redis

Analytics-Module sind Sinks und können so selbst geschrieben werden!

JOBS

Spring Batch / Spring for Hadoop

Vorgefertigte Job-Module

filejdbc

hdfsjdbc

jdbchdfs

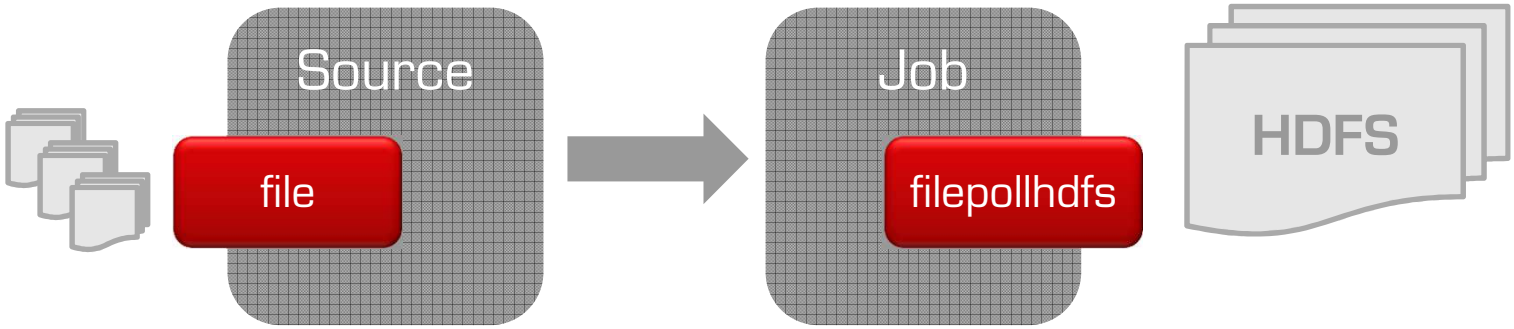
hdfsmongodb

filepollhdfs

Job-Kommandos (XD-Shell)

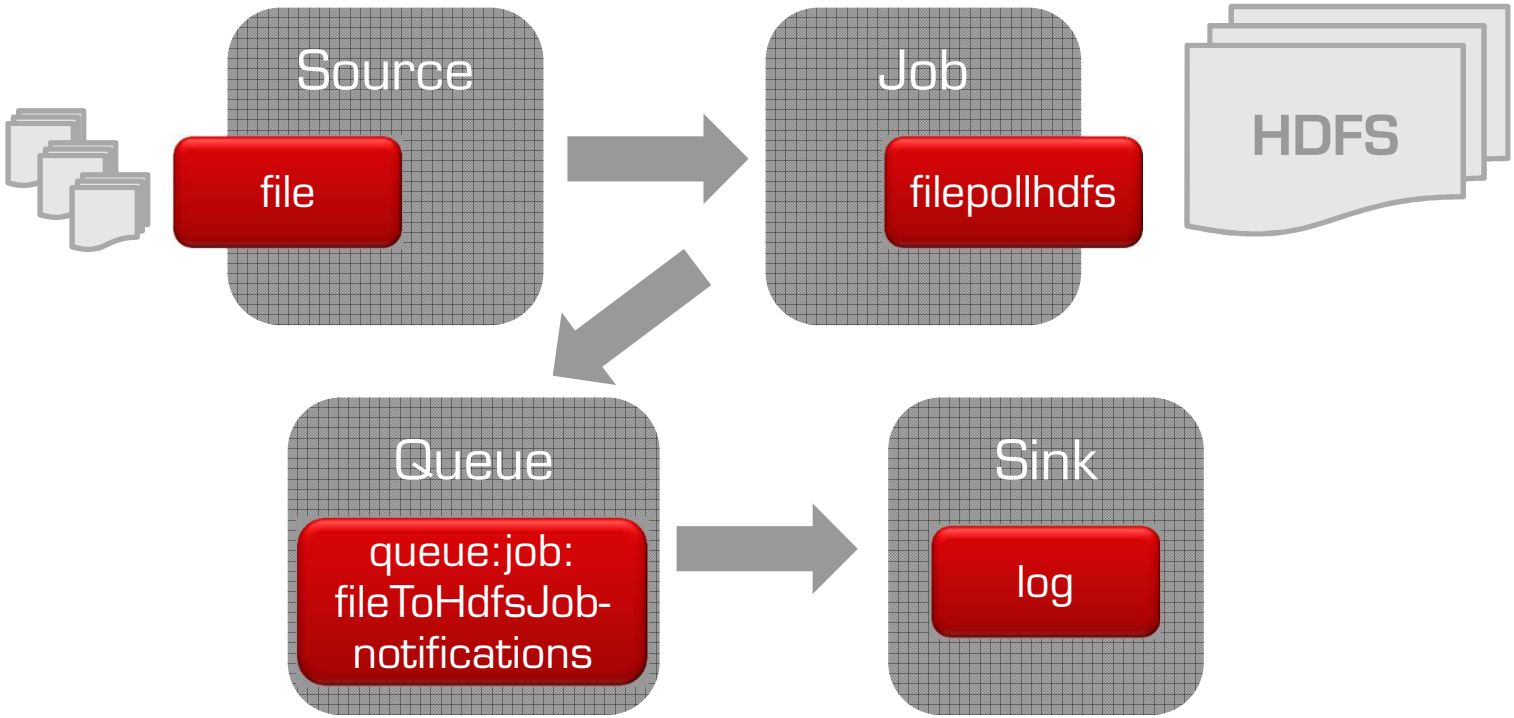
```
xd:> job create myjob –definition "jdbchdfs –sql='select col1,col2,col3 from some_table'"
xd:> job launch myjob
xd:> stream create –name cronStream –definition "trigger –cron='0/5 * * * * *'
        –payload='{"param1":"Kenny"}' > queue:job:myJob"
```

JOBS



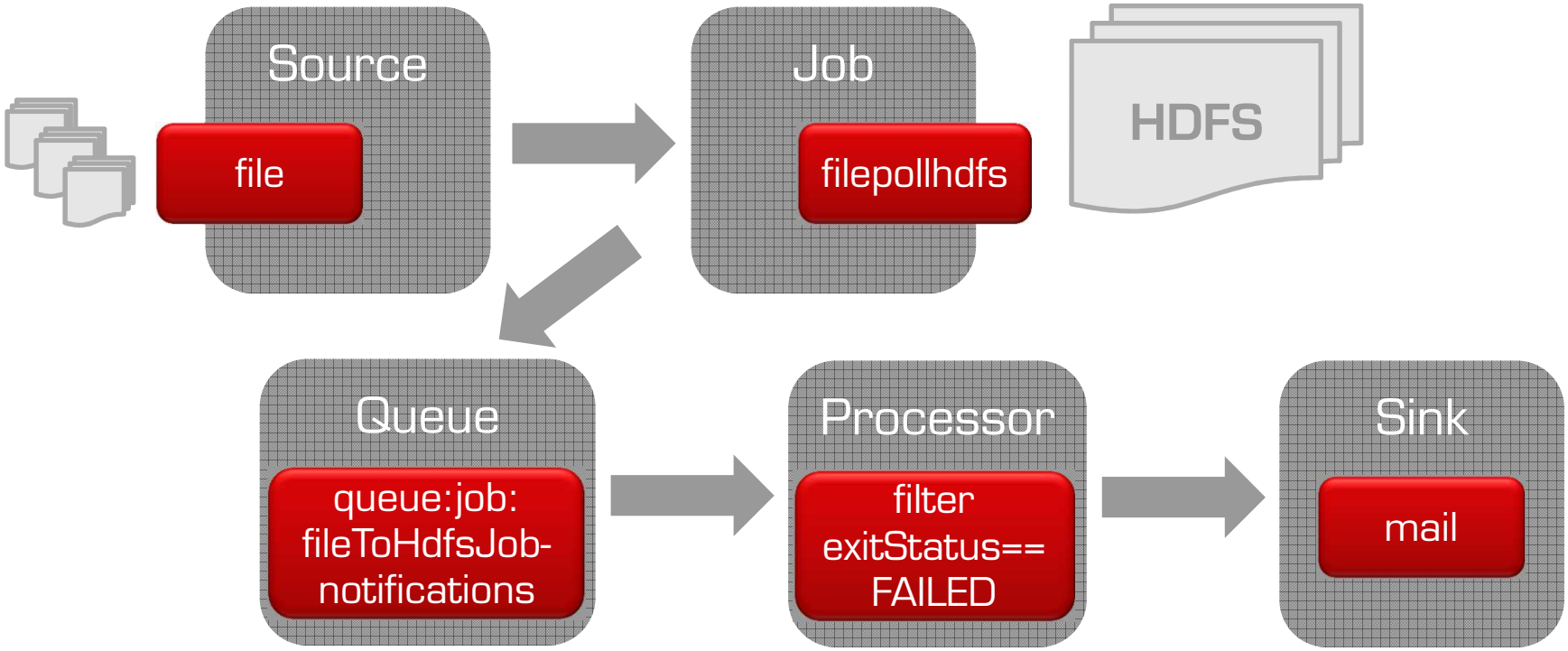

```
xd:> job create fileToHdfsJob –definition "filepollhdfs –names=forename,surname,address"
xd:> stream create csvStream –definition "file –ref=true –dir=/mycsvdir –pattern=*.csv >
        queue:job:fileToHdfsJob"
```

JOBS

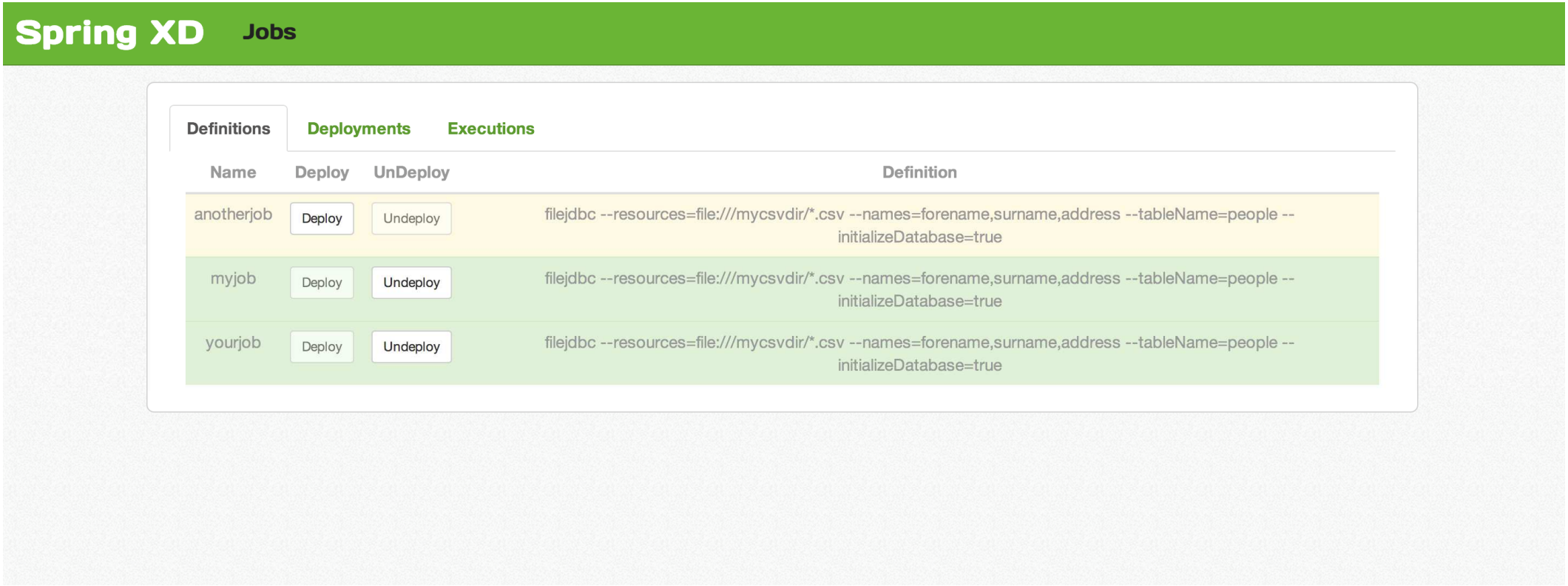


xd:> stream create jobNotifications –definition "queue:job:fileToHdfsJob-notifications > log"

JOBS


Source
file
Job
filepollhdfs
HDFS
Queue
queue:job:
fileToHdfsJob-
notifications
Processor
filter
exitStatus==
FAILED
Sink
mail

JOBS


Spring XD
Jobs
Definitions
Deployments
Executions
Name
Deploy
UnDeploy
Definition
anotherjob
Deploy
Undeploy
filejdbc --resources=file:///mycsvdir/*.csv --names=forename,surname,address --tableName=people --initializeDatabase=true
myjob
Deploy
Undeploy
filejdbc --resources=file:///mycsvdir/*.csv --names=forename,surname,address --tableName=people --initializeDatabase=true
yourjob
Deploy
Undeploy
filejdbc --resources=file:///mycsvdir/*.csv --names=forename,surname,address --tableName=people --initializeDatabase=true

ERWEITERBARKEIT

Source / Processor / Sink: Spring Integration

Job: Spring Batch (Spring for Hadoop)

Einhaltung von Konventionen

- Channel input / output
- Job-Name job

Packaging / Contribution Model: Spring Boot ab Alpha

Classpath-Trennung von Modulen möglich

ERWEITERBARKEIT

Beispiel: Processor

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans (…)>

        <channel id="input" />

        <transformer input-channel="input" output-channel="output">
                <beans:bean class="de.codecentric.xd.LogEntryTransformer" />
        </transformer>

        <channel id="output" />

</beans:beans>
```

FAZIT

Bisher Milestone 5 veröffentlicht

- Interessanter Eindruck

One-Stop-Shop für Big-Data-Anwendungen

- statt Kombination vieler Produkte

Großes Engagement von Pivotal

Interessant in Verbindung mit anderen Frameworks / Produkten

- Hadoop YARN
- CloudFoundry
- Reactor

FRAGEN?

Tobias Flohre
Dennis Schulte

codecentric AG
Merscheider Straße 1
42699 Solingen

tobias.flohre@codecentric.de
dennis.schulte@codecentric.de

www.codecentric.de
blog.codecentric.de
www.meettheexperts.de