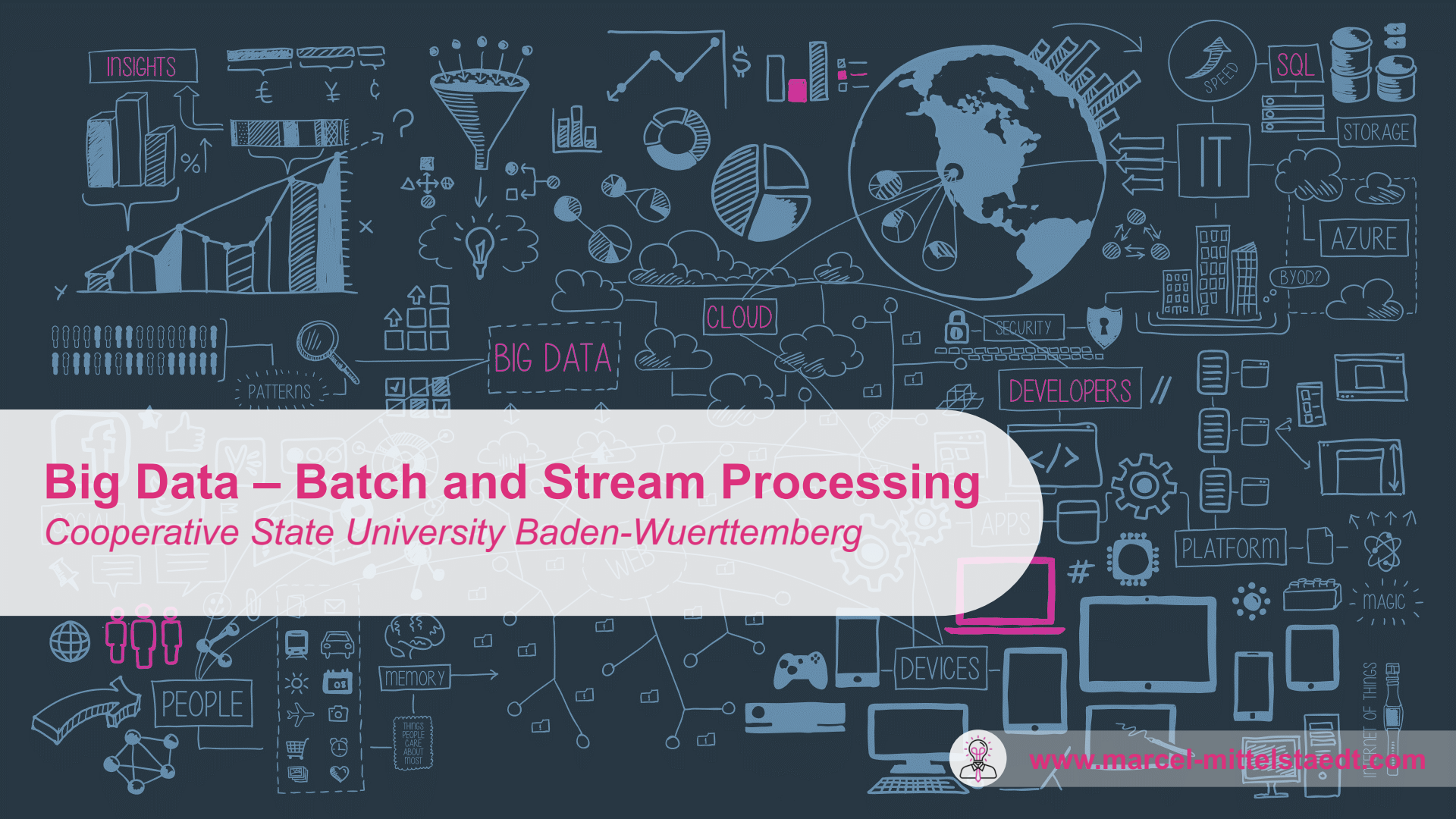
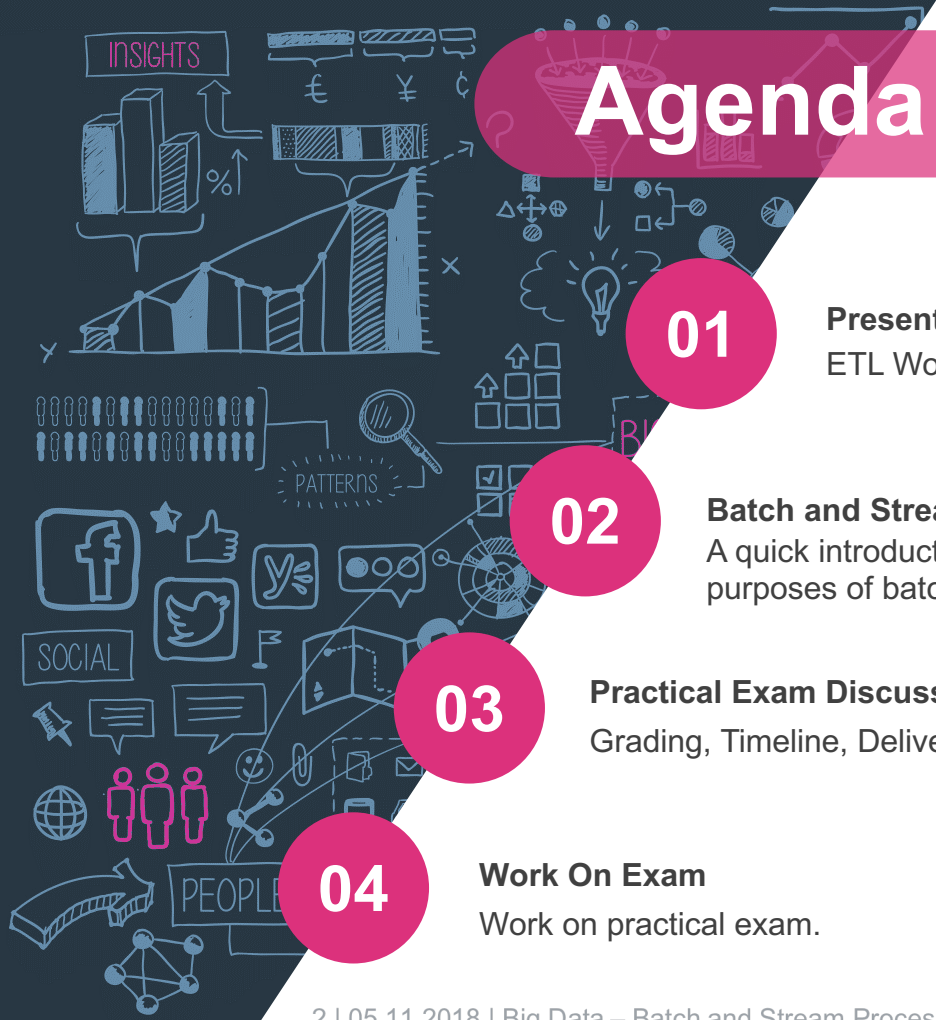


Big Data – Batch and Stream Processing

Cooperative State University Baden-Wuerttemberg

www.marcel-mittelstaedt.com



Agenda – 05.11.2018

01

Presentation and Discussion: Exercise Of Last Lecture

ETL Workflow: Working with PDI on Hadoop and Hive

02

Batch and Stream Processing

A quick introduction to the principles, architectures and purposes of batch and stream processing in terms of big data.

03

Practical Exam Discussion

Grading, Timeline, Deliverables, Presentation Procedure, Exam Topics

04

Work On Exam

Work on practical exam.

Schedule

	Lecture Topic	*HandsOn*
01.10.2018 16:00-19:30 Ro. 0.11	About This Lecture, Introduction to Big Data	Hadoop
08.10.2018 16:00-19:30 Ro. 0.11	(Non-)Functional Requirements Of Distributed Data-Systems, Data Models and Access	MapReduce, Hive, HiveQL
15.10.2018 16:00-19:30 Ro. 0.11	Challenges Of Distributed Data Systems: Replication	Hive/HDFS Partitioning, HiveServer2
22.10.2018 16:00-19:30 Ro. 0.11	Challenges Of Distributed Data Systems: Partitioning	Spark, Scala and PySpark/Jupyter
29.10.2018 16:00-19:30 Ro. 0.11	ETL Workflow And Automation	Pentaho Data Integration
05.11.2018 16:00-19:30 Ro. 0.11	**Batch and Stream Processing**	Work On Practical Examination
12.11.2018 16:00-19:30 Ro. 0.11	Work On Practical Examination	
19.11.2018 16:00-19:30 Ro. 0.11	Presentation Of Practical Examination	

Solution – Exercise 05

ETL Workflow:
Working with PDI on Hadoop and Hive

Solution

Prerequisites:

- Execute all preparation and example tasks of previous HandsOn slides of last lecture
- Start HDFS, YARN and HiveServer2
- Start PDI (spoon.sh or spoon.bat)

Solution

To be added…

Introduction to:
Batch and Stream Processing

A quick introduction to the principles, architectures and purposes of batch and stream processing in terms of big data.

Batch vs Stream Processing

	Batch Processing	Stream Processing
Kind of data:	large, historic	volatile, live, stream
Runtime:	minutes, hours, days	real-time/near-real-time
Re-Execution:	possible	„impossible“

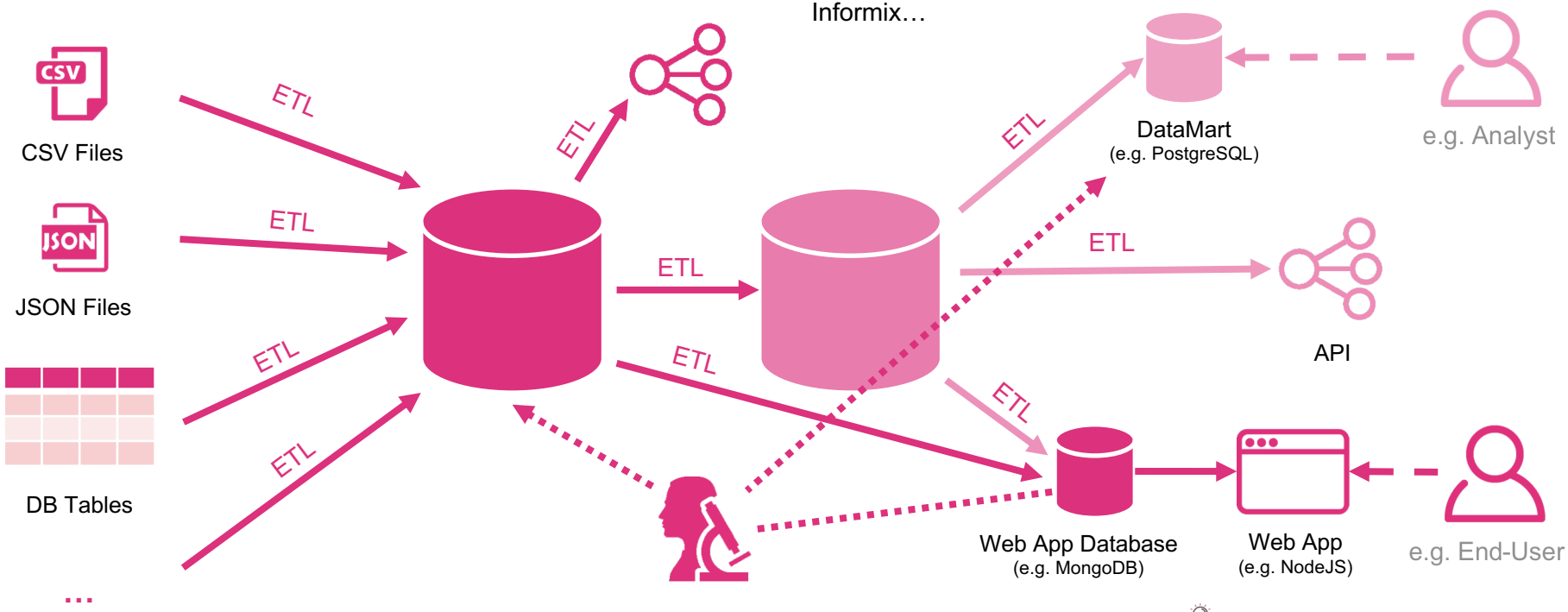
Batch Processing – Example Data Flow

Source Systems:
e.g. CRM, ERP, Webserver Logfiles, …

Data Lake:
e.g. Hadoop HDFS

Data Warehouse:
e.g. PostgreSQL, Oracle, MS SQL Server, DB2, Informix…

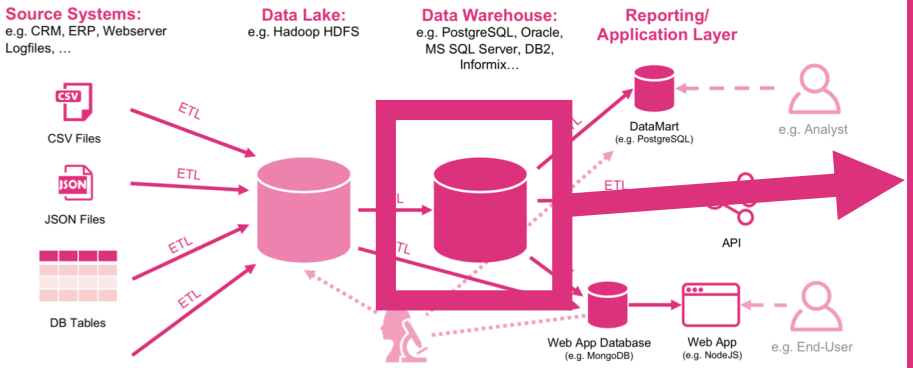
Reporting/ Application Layer

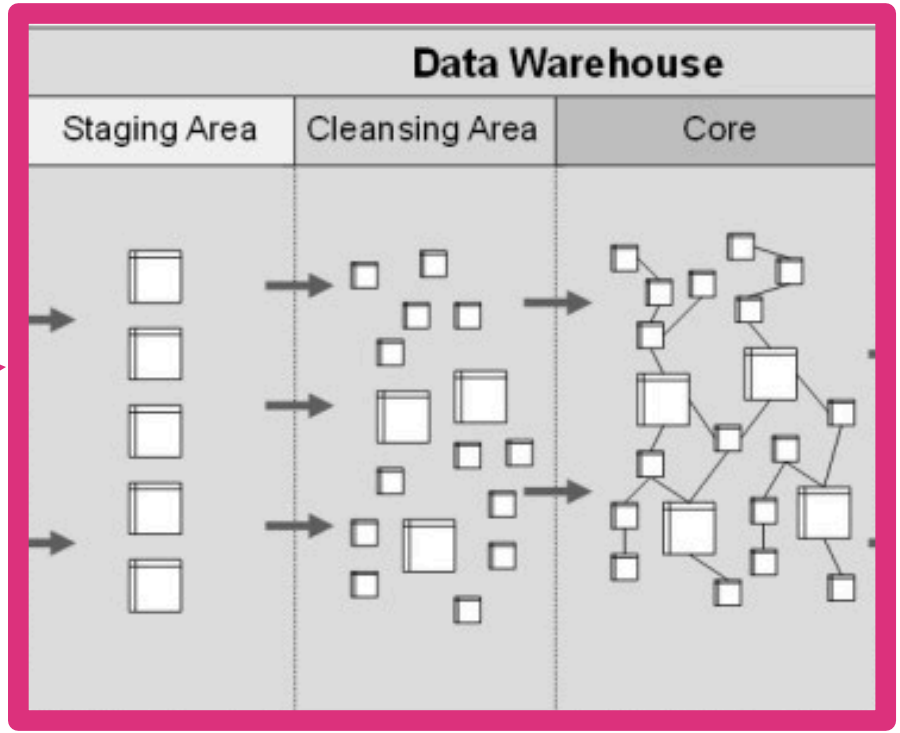


Batch Processing - ETL

ETL (Extract, Transform, Load) *is a basic pattern for data processing, commonly known in data warehousing. It's all about* ***extracting*** *data from a source,* ***transforming*** *the data (e.g. by applying business rules or changing structures) and at the end writing/****loading*** *everything to a target (e.g. Hadoop HDFS, Hive, Relational Database, Data Warehouse, Data Mart etc.).*

Batch Processing – Example Data Flow





Batch Processing – Dissociation Datawarehousing

- a Big Data data-system can be a part or source of a Data Warehouse, e.g.:
  - Data Lake or
  - Enterprise Data Hub

- A Data Warehouse is not Big Data

Data Warehouse

- handles mainly structured data
- suitable for small amounts of data
- focuses on analytical and reporting purposes
- 100% accuracy

Big Data

- handles data in any kind of structure
- serves broad variety of data-driven pur poses (e.g. analytical, data science, data-driven applications, …)
- usually not about 100% accuracy

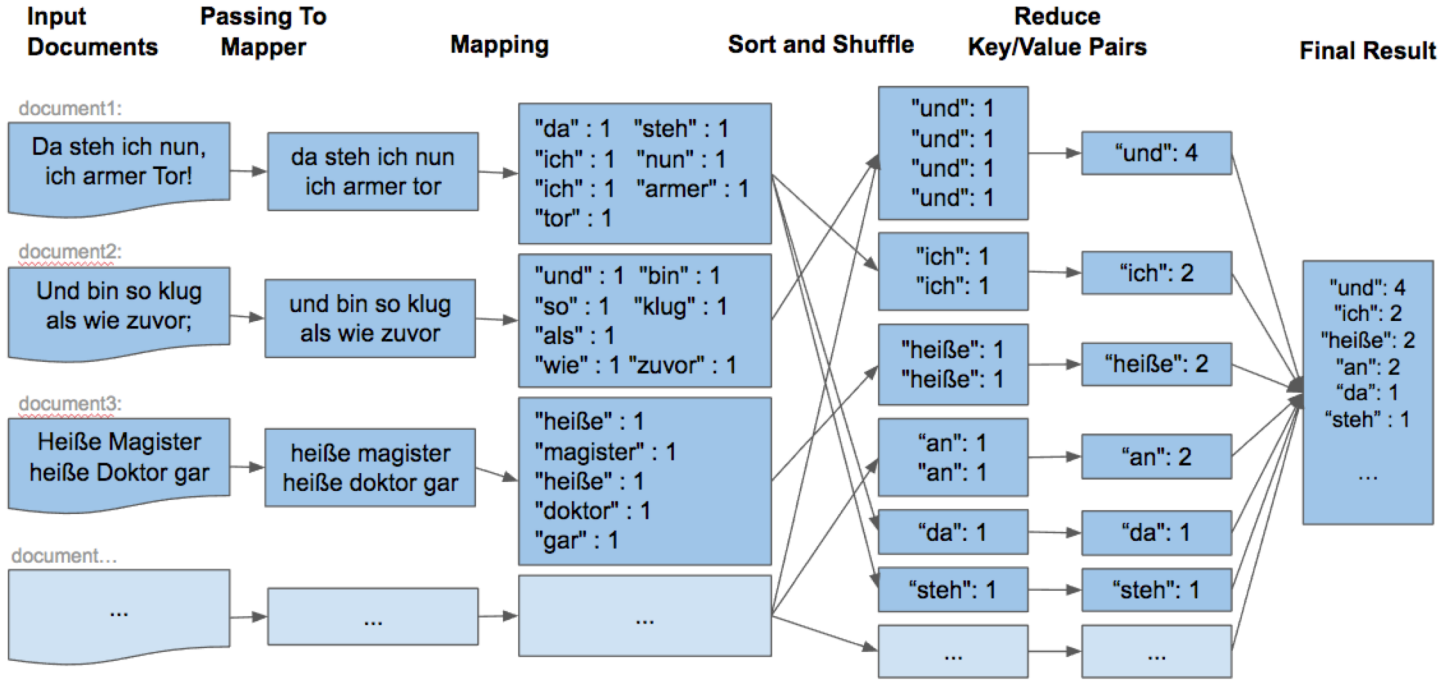
Distributed Batch Processing

Distributed Storage

Distributed Processing

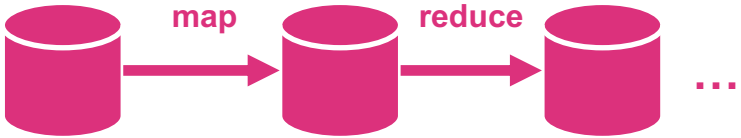
Batch Processing – MapReduce

- Programming paradigm for processing large datasets in parallel on a distributed cluster



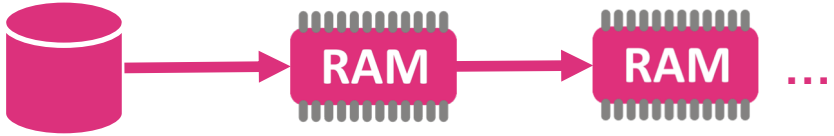
Batch Processing – MapReduce vs Spark

Hadoop MapReduce



- Good reliability
- Bad performance

Spark



- Bad reliability
- Good performance

Spark vs Flink

- **batch processing framework** that emulates stream processing
- **stream processing** = Execution of **micro batches**
- cuts event stream into micro batches and processes each batch
- **Latency:** (several) seconds
- **Maturity:** high, many libraries, huge community and developer base

- **stream processing framework** that emulates batch processing
- **batch processing** = bounded stream processing
- processes each event when it arrives
- **Latency:** milliseconds-seconds
- **Maturity:** medium, limited libraries, medium community and developer base

Spark vs Flink

- good choice for batch processing
- stream processing:

if **reliabilty > latency**

- good choice for stream processing:

if **latency > reliability**

Stream Processing – **Definition**

Data Stream:

- is data made available over time in an incemental way
- created by:
  - static data (e.g. file or database read line-wise)
  - dynamic data (e.g. logs, sensors, function calls, …)

Event:

- is an immutable record/item in a stream
- usually represented and encoded in e.g. JSON, XML, CSV or binary encoded
- pendants

Stream Processing – **Use Cases/Data Sources**

- User Interaction, e.g.:
  - webserver logfiles
  - tracking data like Google Analytics
  - recommendation systems (advertisement, personalization)
  - …
- Sensor data
- Any API serving data in a stream, e.g.:
  - Twitter Postings API,
  - Message Queue/Broking Systems (e.g. Kafka, ZeroMQ, RabbitMQ…)
  - IoT
- Location Tracking (e.g. DriveNow, Car2Go, Google Maps…)
- …

Message Broker/Queues

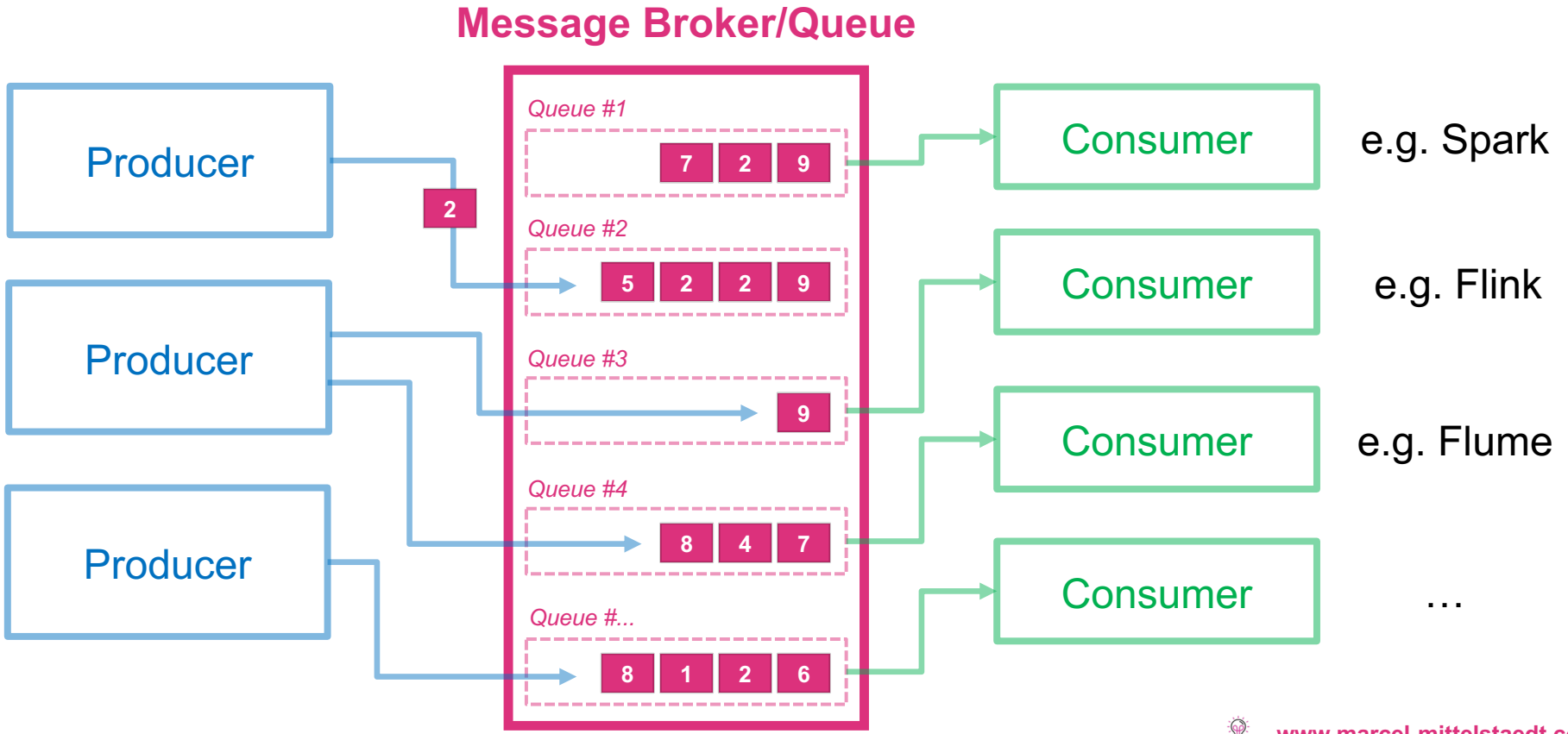
Why:

- decouples Producer and Receiver
- maintains queues for different topics
- temporarily persists messages
- Notifies subscribers on new messages
- Micro Services, IoT…

Examples:

- RabbitMQ,
- ZeroMQ,
- Kafka,
- ActiveMQ,
- Kestrel,
- …

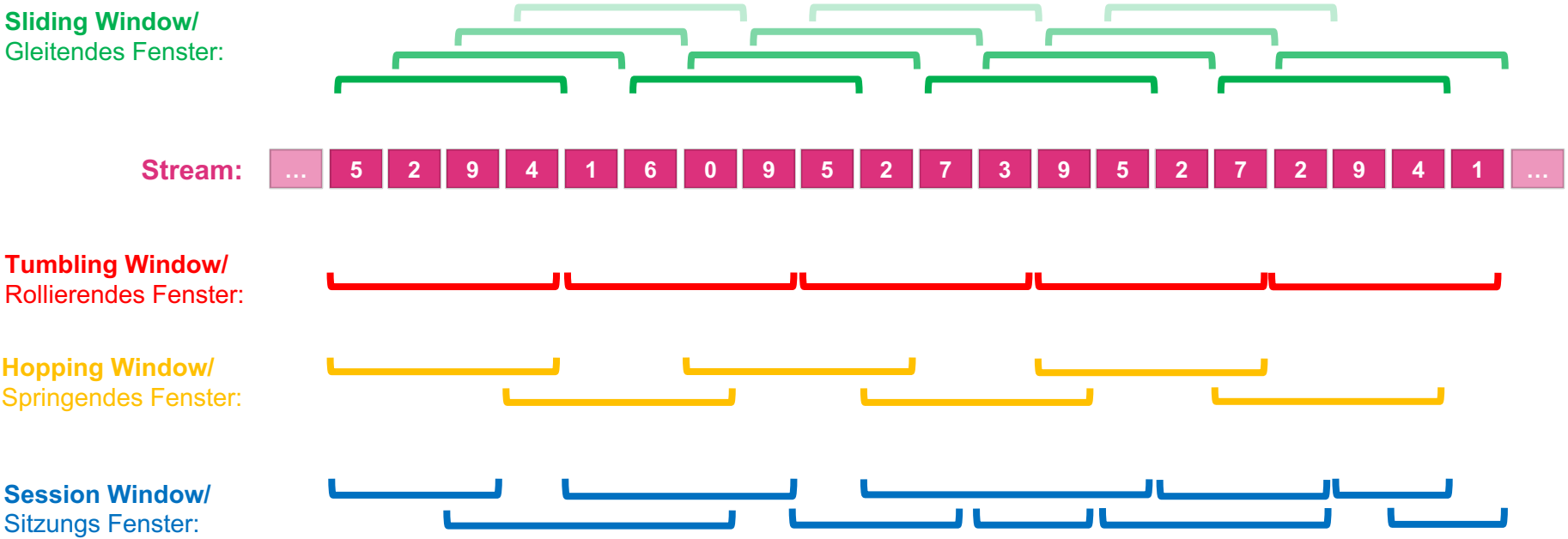
Message Broker/Queues



Message Broker/Queues vs Data-Systems

	Message Broker/Queue	Database
Persistence:	- **temporarily** (delete once transmitted or after a certain timeframe)	- **persistent**
Data Retrieval:	- **Subscription-based** (even if e.g. Kafka supports qu ery-based)	- **Query-based** (execute query to receive data)
Communi-cation:	- **Initiated by MB**	- **Initiated by clients**

Stream Processing – Windows

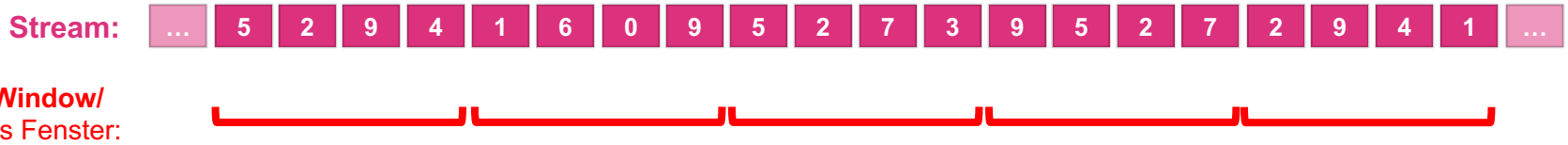


Stream Processing – Tumbling Windows

Data Stream:
- Fixed length
- Non-overlapping
- An event relates to exactly one window

Twitter Example:
- Calculate the count of tweets for every 4 seconds

```
SELECT count(*) as tweet_count FROM Twitter_Stream TIMESTAMP BY
CreationTime GROUP BY TumblingWindow(second, 4)
```

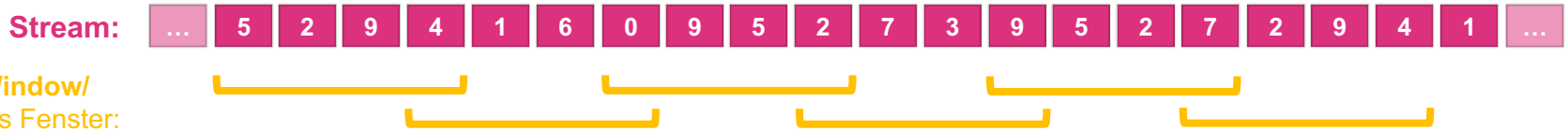


Stream Processing – Hopping Windows

Data Stream:
- Fixed length
- Overlapping (with fix steps)
- An event relates to more than one window

Twitter Example:
- Every 3 seconds calculate the count of tweets for last 4 seconds

```
SELECT count(*) as tweet_count FROM Twitter_Stream TIMESTAMP BY
CreationTime GROUP BY HoppingWindow(second, 4, 3)
```



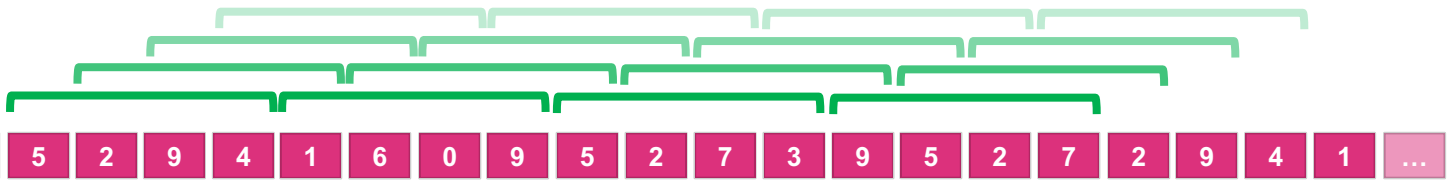
Stream Processing – Sliding Windows

Data Stream:
- Fixed length
- Overlapping (dependending on event)
- An event can relate to more than one window
- Event-based, every window contains at least one event and is continuesly slided forward

Twitter Example:
- Calculate count of tweets which have used a certain hashtag. Count number of tweets including hashtags, which were used more than 10 times within the last 4 seconds

```
SELECT HashTag, count(*) as tweet_count FROM Twitter_Stream TIMESTAMP BY
CreationTime GROUP BY Hashtag, SlidingWindow(second, 4)HAVING count(*) > 10
```

Sliding Window/
Gleitendes Fenster:

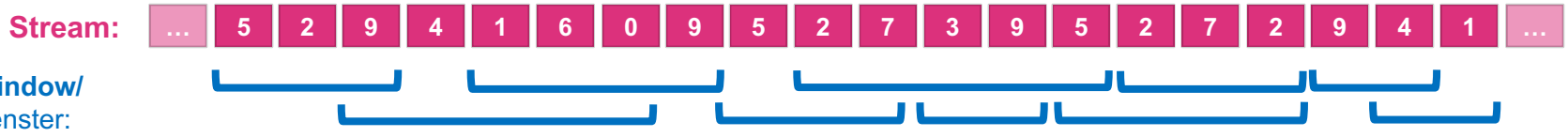


Stream Processing – Session Windows

Data Stream:
- Arbitrary-length
- Overlapping possible
- An event can relate to more than one window
- Windows containing no events are filtered out
- Fix start event (e.g. login) and end event (logout or timeout)
- Max size usually limited

Twitter Example:
- Calculate count of tweets for certain hashtags that occur within less than 5 minutes to each other

```
SELECT HashTag, count(*) as tweet_count FROM Twitter_Stream TIMESTAMP BY
CreationTime GROUP BY HashTag, SessionWindow(minute, 5)
```



Word Count on Stream – Apache Spark

```
// Initialize Spark Session
val spark = SparkSession
  .builder()
  .master("local")
  .appName("Socket_Streaming")
  .getOrCreate()

// Initialize Socket Stream
val socketStreamDf = spark.readStream
  .format("socket")
  .option("host", "localhost")
  .option("port", 4711)
  .load()

// parse, group, window and aggregate data
val words = socketStreamDf.as[String].flatMap(_.split(" "))
val wordCounts = words.groupBy("value").count()
val query = wordCounts.writeStream
  .outputMode("complete")
  .format("console")
  .start()

query.awaitTermination()
```

Input could also be **Kafka, Flume, Kinesis, Sockets, File, Custom Connectors…**

Output could also be **HDFS, Kafka, Kinesis, Console, …**

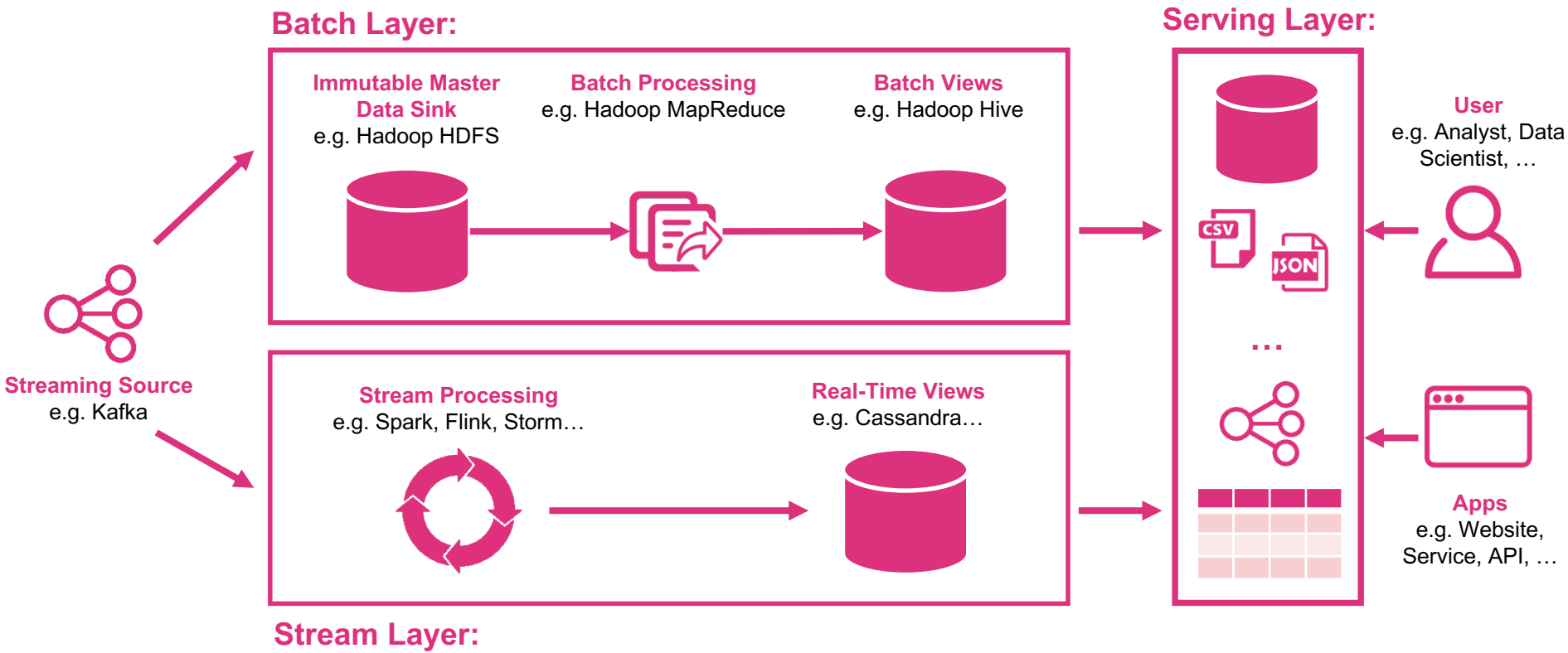
Word Count on Stream – Apache Flink

```
// Initiate Execution Environment
val env = StreamExecutionEnvironment.getExecutionEnvironment

// Read data from socket on localhost, port: 4711
val text = env.socketTextStream("localhost", 4711, '\n')

// parse, group, window and aggregate data
val word_counts = text
  // Split lines into 2-tuples: (word,1)
  .flatMap(_.toLowerCase.split("\\s"))
  .filter(_.nonEmpty)
  .map((_, 1))
  // group by word (tuple field "0")
  .keyBy(0)
  // sliding window: 5 second length, 1 second trigger interval
  .timeWindow(Time.seconds(5), Time.seconds(1))
  // sum up word count (tuple field "1")
  .sum(1)

// print result (single thread)
word_counts.print().setParallelism(1)
}

// Execute
env.execute("Streaming Example: WordCount")
```

Input Could also be **Kafka, Kinesis, RabbitMQ, NiFi, ActiveMQ, Socket, File, Netty, Custom Connectors…**

Output could also be **HDFS, Kafka, Elasticsearch, Kinesis, RabbitMQ, NiFi, Akka, Redis, Cassandra, Flume, ActiveMQ, Custom Connectors… …**

Batch&Stream Processing – Lambda Architecture



Practical Exam

Topics, Deliverables, Presentation…

Practical Exam - Grading

Grade/Percentage:

- grade will be mixed with grade of another lecture
- therefore, the rating won‘t be a grade (1-6) but a percentage:

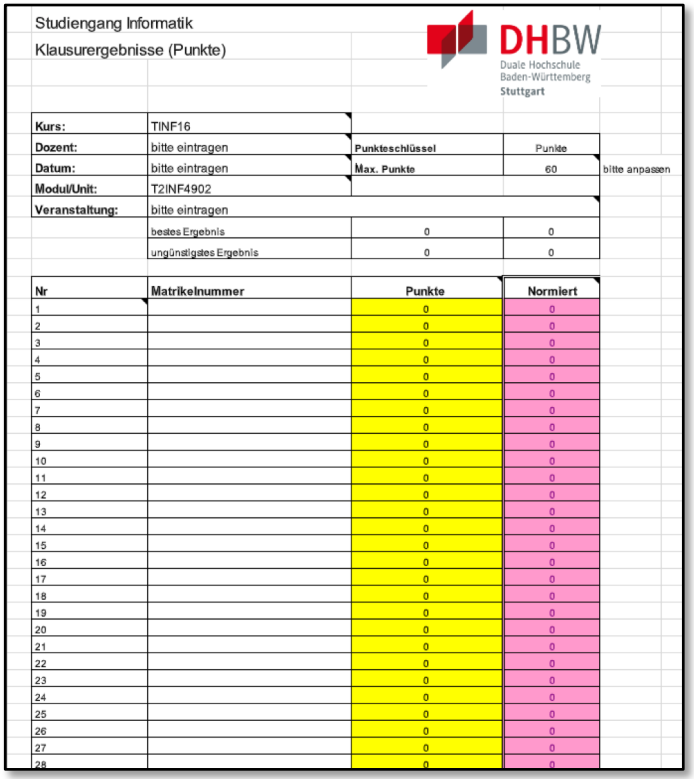
Percentage	Grade
100%	1.0
…	…
50%	4.0
…	…

Timeline:

16.11.2018 18:00 All deliverables (next slide) will be delivered to following email address (DropBox, Google-Drive…):

contact@marcel-mittelstaedt.com

19.11.2018 16:00-19:30 Presentation of Practical Exam



Studiengang Informatik

Klausurergebnisse (Punkte)

DHBW Duale Hochschule Baden-Württemberg Stuttgart

Kurs:	TINF16			
Dozent:	bitte eintragen	Punkteschlüssel	Punkte	
Datum:	bitte eintragen	Max. Punkte	60	bitte anpassen
Modul/Unit:	T2INF4902			
Veranstaltung:	bitte eintragen			
	bestes Ergebnis	0	0	
	ungünstigstes Ergebnis	0	0	

Nr	Matrikelnummer	Punkte	Normiert
1		0	0
2		0	0
3		0	0
4		0	0
5		0	0
6		0	0
7		0	0
8		0	0
9		0	0
10		0	0
11		0	0
12		0	0
13		0	0
14		0	0
15		0	0
16		0	0
17		0	0
18		0	0
19		0	0
20		0	0
21		0	0
22		0	0
23		0	0
24		0	0
25		0	0
26		0	0
27		0	0
28		0	0

Practical Exam - **Deliverables**

Deliverables:

- A simple Documentation:
  - Explanation of Whole ETL Workflow
  - List of Jobs/Transformations
  - Short description of the purpose of each job/transformation and applied busines rules and transformations
- all PDI Jobs, Transformations and related files (ktr, kjb, kettle.properties, shared.xml… files)
- All Scripts (e.g. Download) or other external applications called within PDI
- All DDLs (`CREATE Table…`):
  - One file for each table
  - Table name = File Name, e.g.:

- Depending on Exam Type:
  - Code of Frontend Application and related Database (DDLs) or
  - Calculated KPIs

Practical Exam - **Presentation**

Procedure:

1. Start ETL Workflow

2. During execution:
    - Quickly explain data source
        - API
        - Data Structure
        - Approach for gathering data
    - Quickly Explain whole ETL Workflow
        - Explain Idea and purpose of each Job/Transformation
        - External ressources/scripts (e.g. download)
        - Explain Data Model (Raw Layer, Final Layer, simple Frontend)

3. After execution:
    - Depending on Exam:
        - Demo of simple Frontend application or
        - Explanation of calculated KPIs

Work On Practical Exam

Time to work on practical exam

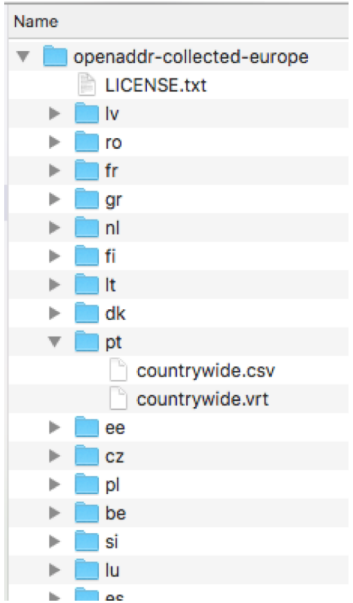
Use OpenAdresses Data To Validate Adresses

Practical Exam

Goal

OpenAddresses.io provides regulary exports of worldwide adresses (we will focus on europe for now):

- http://results.openaddresses.io/
- https://s3.amazonaws.com/data.openaddresses.io/openaddr-collected-europe.zip




```
LON,LAT,NUMBER,STREET,UNIT,CITY,DISTRICT,REGION,POSTCODE,ID,HASH
13.1688262,52.5078776,9,Potsdamer Chaussee,,Berlin,,,13593,,5ef12a3087c99461
13.1689364,52.507843,9 A,Potsdamer Chaussee,,Berlin,,,13593,,a9f40456e296699a
13.1692019,52.5078327,9 B,Potsdamer Chaussee,,Berlin,,,13593,,b6ece1ead310fc9f
13.1693468,52.507773,9 C,Potsdamer Chaussee,,Berlin,,,13593,,839a7d07f6e663ae
13.1694613,52.5077415,9 D,Potsdamer Chaussee,,Berlin,,,13593,,ccfb7c1285bcd45a
13.1698781,52.5076352,11,Potsdamer Chaussee,,Berlin,,,13593,,b86e62e6fb76b33a
13.2591722,52.4354736,1,Potsdamer Straße,,Berlin,,,14163,,31017dd3e09e6930
13.2589704,52.4355088,2,Potsdamer Straße,,Berlin,,,14163,,6516a41eb899cd75
13.2577856,52.4357575,3,Potsdamer Straße,,Berlin,,,14163,,edaee5c1994b3281
13.2574511,52.4358362,4,Potsdamer Straße,,Berlin,,,14163,,9b87f7456c37c259
13.2570002,52.4359144,6,Potsdamer Straße,,Berlin,,,14163,,b2759ff7fe5a960f
13.2570961,52.4363525,7,Potsdamer Straße,,Berlin,,,14163,,5a41311f751606f3
13.2562084,52.436783,7 A,Potsdamer Straße,,Berlin,,,14163,,b86af65860315c1c
13.255622,52.4359221,7 B,Potsdamer Straße,,Berlin,,,14163,,b47315f8ba8452b6
13.2550108,52.4358296,8,Potsdamer Straße,,Berlin,,,14163,,cdc7b606fcbf62e1
…
```

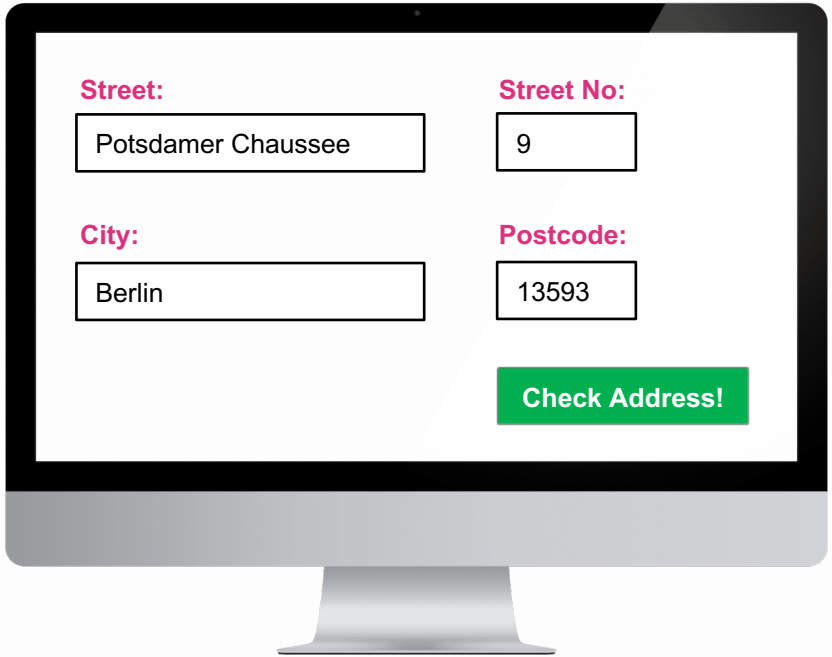
berlin.csv

Goal

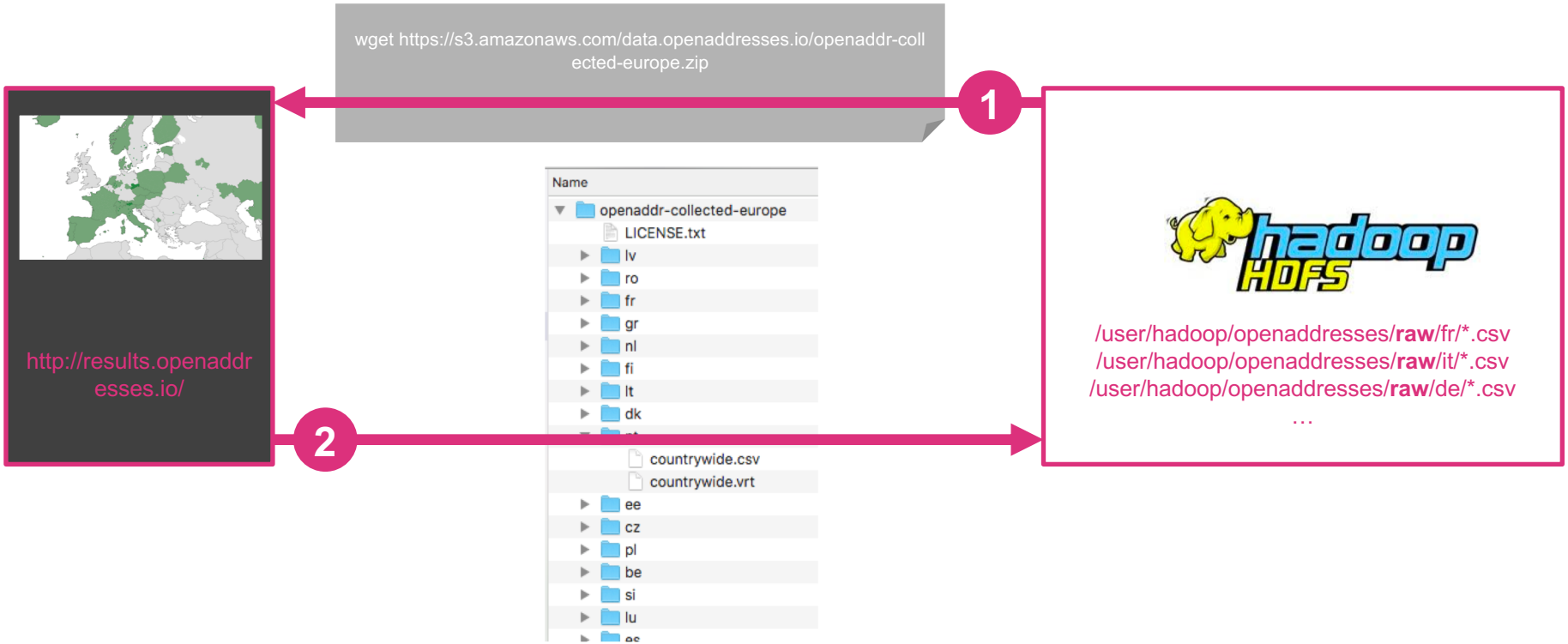
We want to make use of this data to validate adresses entered on a website, to check whether they are real or not.

Workflow:

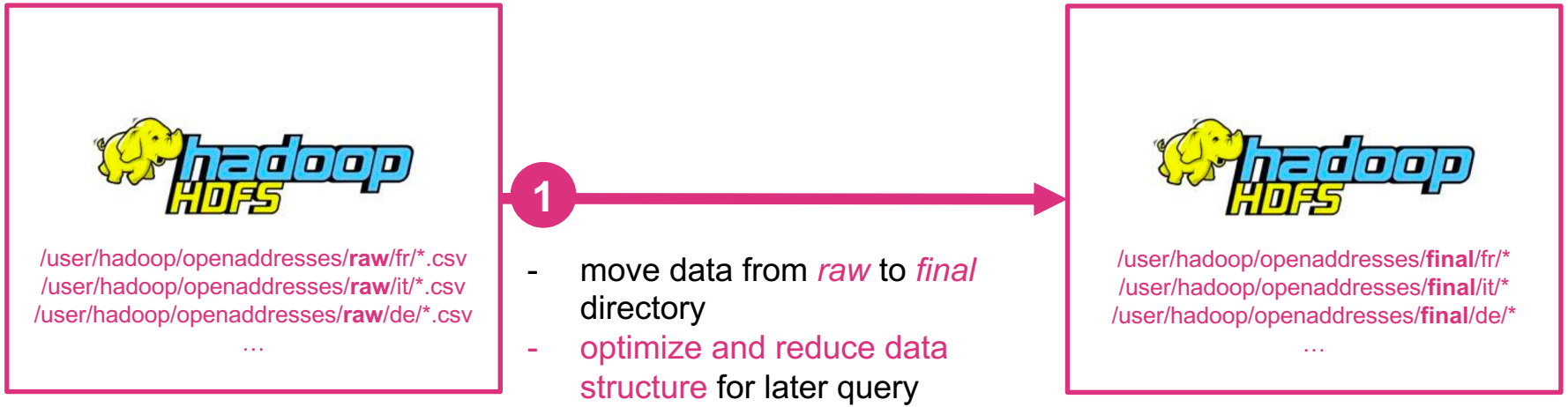
- **Gather data** from OpenAddresses.io
- **Save raw data** (*CSV files*) to HDFS (partitioned by country
- shortcut, e.g. *de*, *fr*, *it*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** address data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Street, City, Postcode…)
  - validate user input against OpenAddress data in end-user database
  - Display result (real or non real address)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



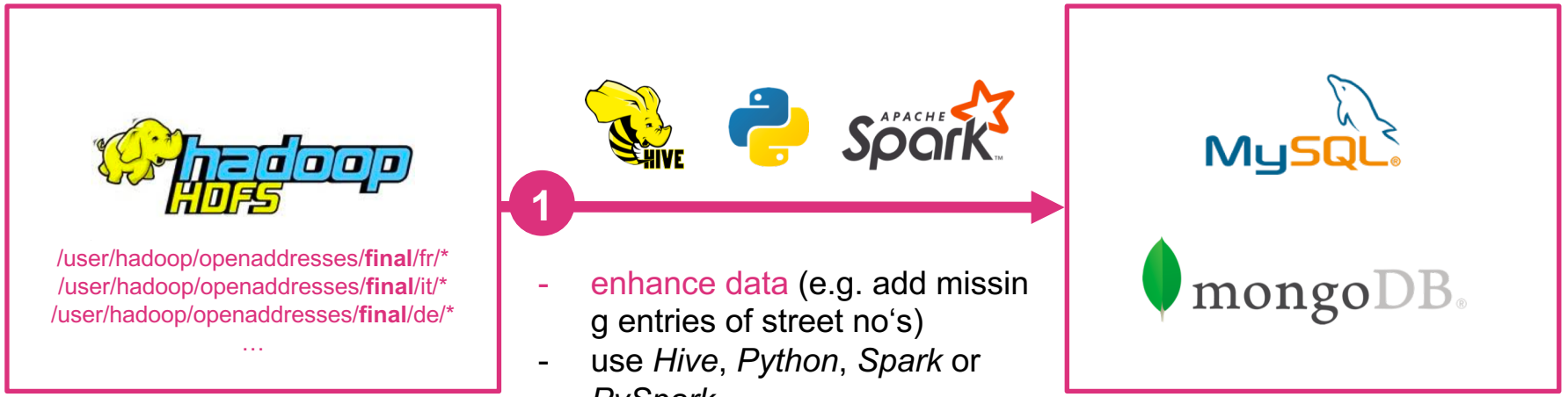
Dataflow: 1. Get Address Data



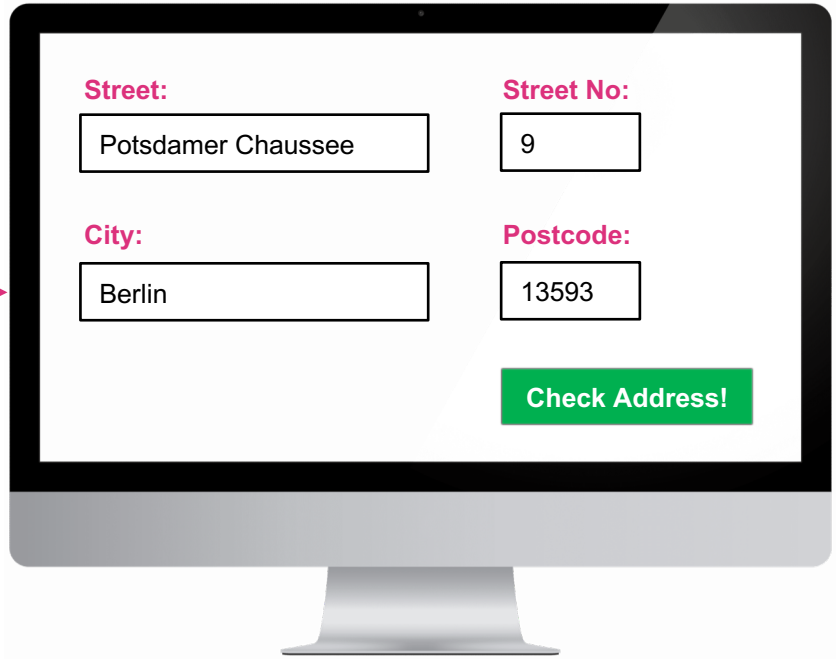
Dataflow: 2. Raw To Final Transfer



Dataflow: 3. Enhance Data And Save Results



Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Street, City, Postcode…)
  - validate user input against OpenAddress data in end-user database
  - Display result (real or non real address)

Use OpenCellID Data To Estimate GSM/UMTS/LTE Coverage

Practical Exam

Goal

OpenCellID.com provides regulary exports of worldwide cell data:

- https://www.opencellid.org
- Latest Full Dump and Diffs: https://www.opencellid.org/downloads.php

Full Database

- cell_towers.csv.gz
  Updated: 2018-10-04 (907MB)

Differential

- OCID-diff-cell-export-2018-10-04-T000000.csv.gz (1385KB)
- OCID-diff-cell-export-2018-10-03-T000000.csv.gz (2350KB)
- OCID-diff-cell-export-2018-10-02-T000000.csv.gz (2293KB)
- OCID-diff-cell-export-2018-10-01-T000000.csv.gz (969KB)
- OCID-diff-cell-export-2018-09-30-T000000.csv.gz (1306KB)
- OCID-diff-cell-export-2018-09-29-T000000.csv.gz (2279KB)
- OCID-diff-cell-export-2018-09-28-T000000.csv.gz (2378KB)

```
radio,mcc,net,area,cell,unit,lon,lat,range,samples,changeable,created,updated,averageSignal
UMTS,262,2,801,86355,0,13.285512,52.522202,1000,7,1,1282569574,1300155341,0
GSM,262,2,801,1795,0,13.276907,52.525714,5716,9,1,1282569574,1300155341,0
GSM,262,2,801,1794,0,13.285064,52.524,6280,13,1,1282569574,1300796207,0
UMTS,262,2,801,211250,0,13.285446,52.521744,1000,3,1,1282569574,1299466955,0
UMTS,262,2,801,86353,0,13.293457,52.521515,1000,2,1,1282569574,1291380444,0
UMTS,262,2,801,86357,0,13.289106,52.53273,2400,3,1,1282569574,1298860769,0
UMTS,262,3,1107,83603,0,13.349675,52.497575,3102,222,1,1282672189,1300710809,0
GSM,262,2,776,867,0,13.349711,52.497367,1000,214,1,1282672189,1301575206,0
GSM,262,3,1107,13971,0,13.349743,52.497437,1000,212,1,1282672189,1300710809,0
GSM,262,3,1107,355,0,13.34963,52.497378,1000,198,1,1282672189,1300710809,0
UMTS,262,3,1107,329299,0,13.349223,52.497519,3041,186,1,1282672189,1299860879,0
```

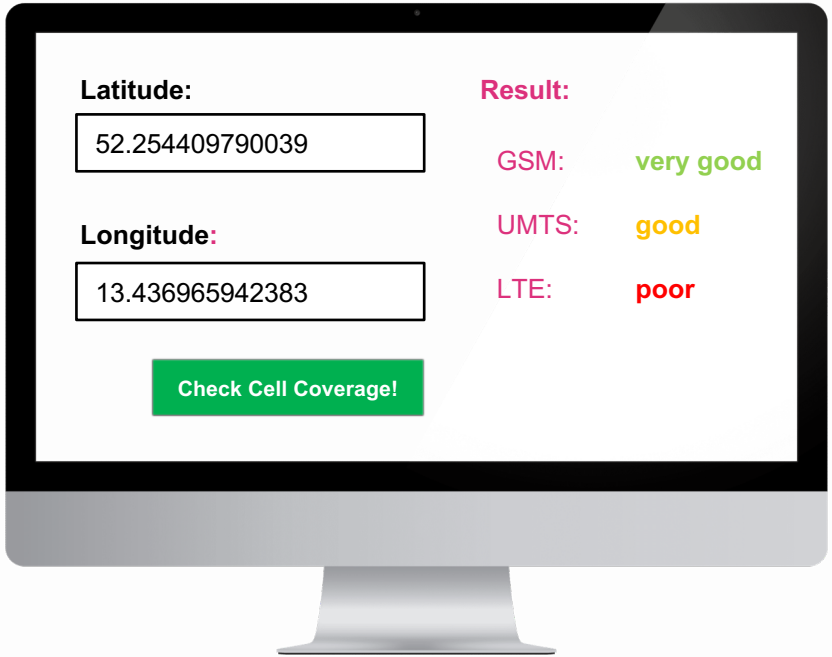
cell_towers.csv

Goal

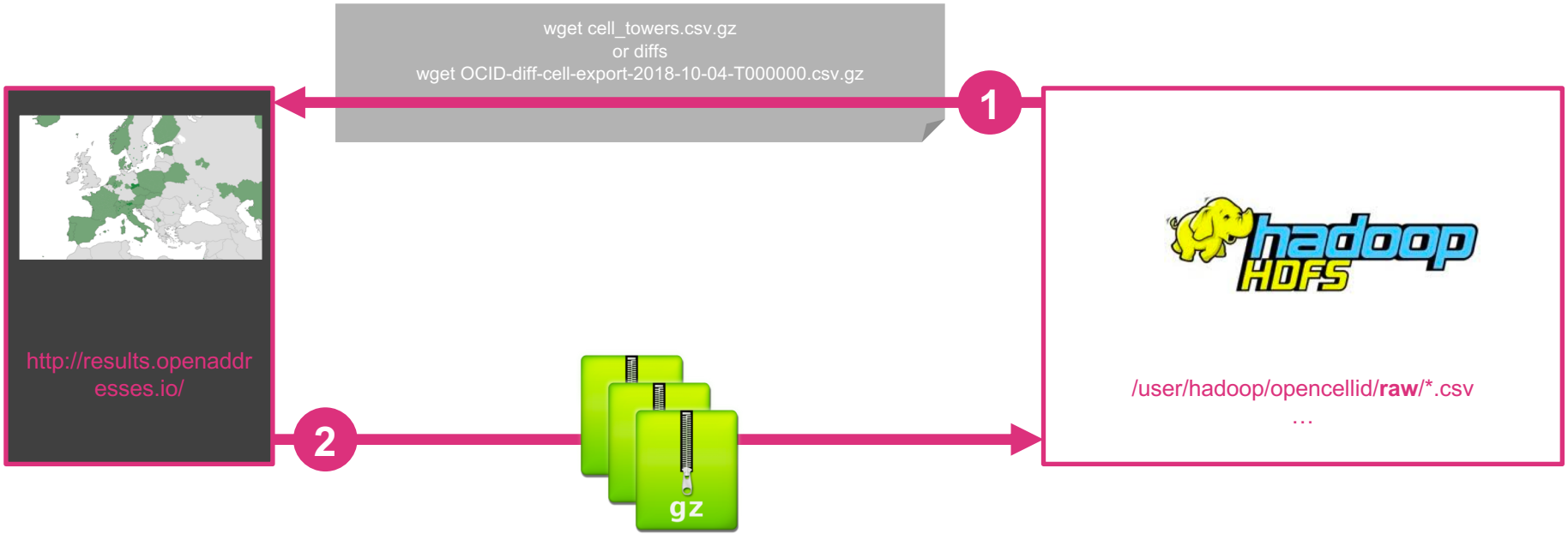
We want to make use of this data to estimate the coverage of GSM, UMTS and LTE for a certain place (*latitude, longitude*).

Workflow:

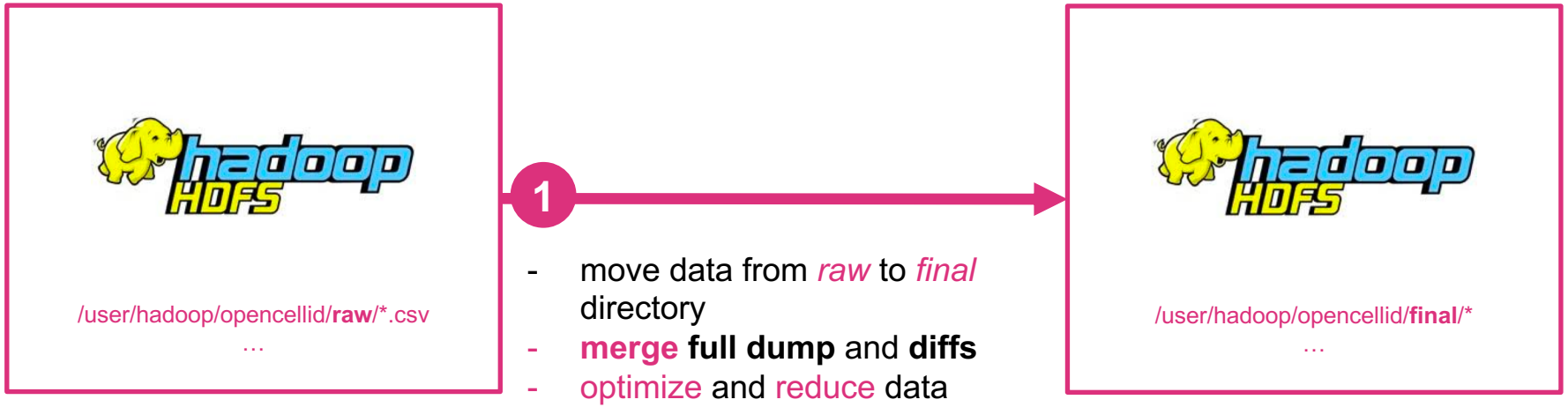
- **Gather data** from OpenCellID.com
- **Save raw data** (*CSV files*) to HDFS (partitioned by radio, e.g. *GSM*, *UMTS*, *LTE*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** address data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Latitude, Longitude…)
  - checks against OpenCellID data in end-user database
  - Display result (GSM, LTE and UMTS coverage)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



Dataflow: 1. Get Cell Data

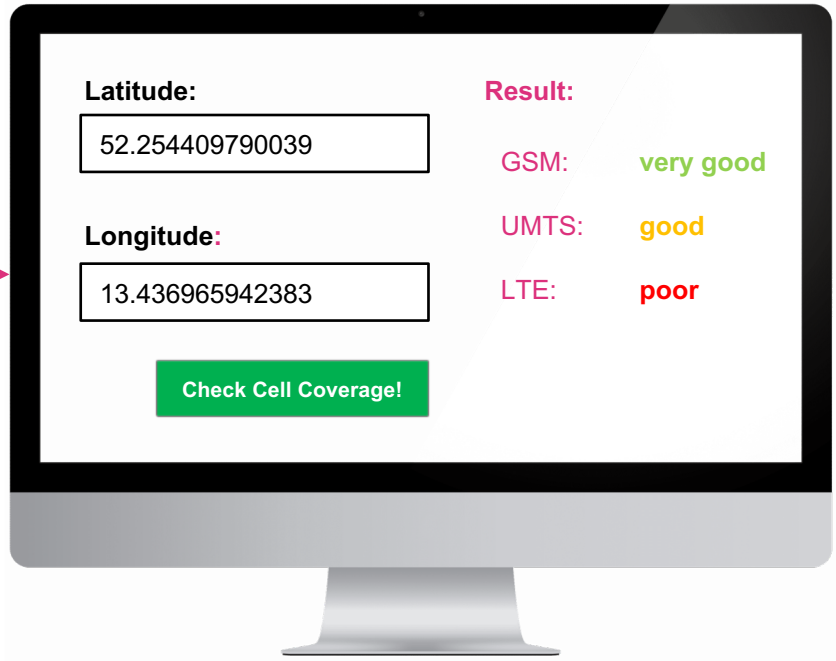


Dataflow: 2. Raw To Final Transfer



Dataflow: 3. Enhance Data And Save Results

Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (Latitude, Longitude…)
  - checks against OpenCellID data in end-user database
  - Display result (GSM, LTE and UMTS coverage)

Use *kaggle.com* Hubway Data To Calculate Bike Sharing Usage KPIs

Practical Exam

Goal

kaggle.com provides monthly exports of Hubway bike sharing trip records:

- https://www.bluebikes.com/
- Latest Full Dumps: https://www.kaggle.com/acmeyer/hubway-data

```
"tripduration","starttime","stoptime","start station id","start station name","start station latitude","start station longitude","end station id","end station name","end station latitude","end station longitude","bikeid","usertype","birth year","gender"
"133","2015-12-01 00:01:52","2015-12-01 00:04:06","9","Agganis Arena - 925 Comm Ave.","42.351246","-71.115639","41","Packard's Corner - Comm. Ave. at Brighton Ave.","42.352261","-71.123831","199","Customer","1995","1"
"1522","2015-12-01 00:05:30","2015-12-01 00:30:53","41","Packard's Corner - Comm. Ave. at Brighton Ave.","42.352261","-71.123831","54","Tremont St / West St","42.354979","-71.063348","876","Customer","1983","1"
"153","2015-12-01 00:07:46","2015-12-01 00:10:20","75","Lafayette Square at Mass Ave / Main St / Columbia St","42.36346469304347","-71.10057324171066","67","MIT at Mass Ave / Amherst St","42.3581","-71.093198","757","Subscriber","1995","1"
"435","2015-12-01 00:07:48","2015-12-01 00:15:04","68","Central Square at Mass Ave / Essex St","42.36507","-71.1031","29","Innovation Lab - 125 Western Ave. at Batten Way","42.363732","-71.124565","853","Subscriber","1988","1"
"1208","2015-12-01 00:12:15","2015-12-01 00:32:23","36","Boston Public Library - 700 Boylston St.","42.349673","-71.077303","110","Harvard University Gund Hall at Quincy St / Kirkland S","42.376369","-71.114025","437","Customer","1982","1"
"1117","2015-12-01 00:16:31","2015-12-01 00:35:09","31","Seaport Hotel","42.348833","-71.041747","67","MIT at Mass Ave / Amherst St","42.3581","-71.093198","1161","Subscriber","1988","1"
"1287","2015-12-01 00:16:50","2015-12-01 00:38:18","10","B.U. Central - 725 Comm. Ave.","42.350406","-71.108279","23","Mayor Martin J Walsh - 28 State St","42.35892","-71.057629","565","Subscriber","1966","1"
```

Goal

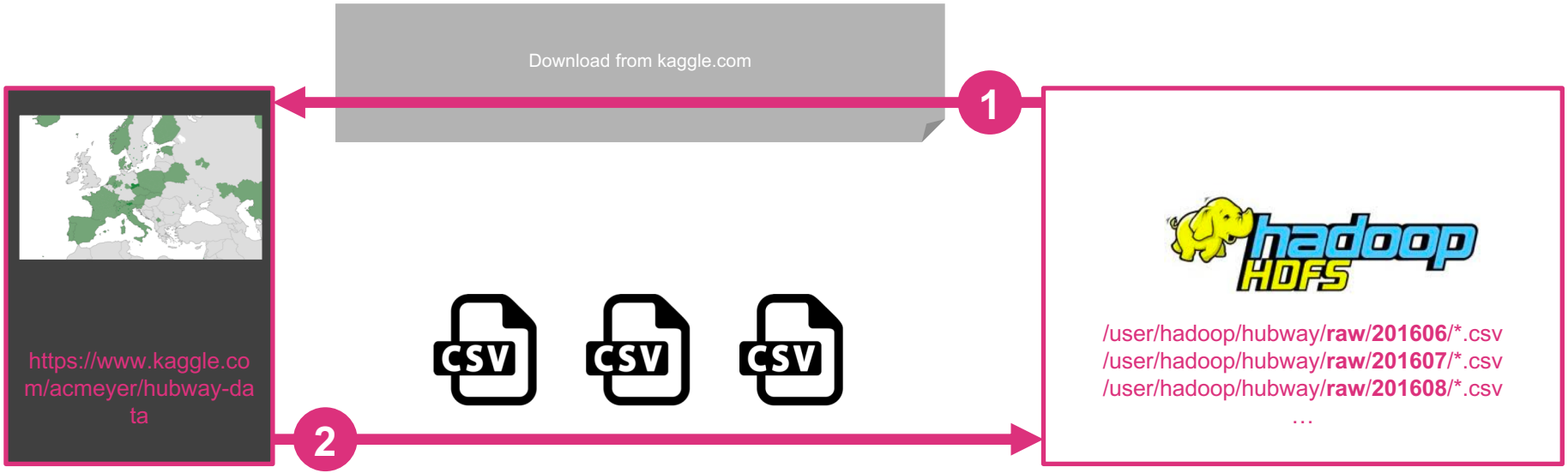
We want to make use of this data to calculate some Usage KPIs.

Workflow:

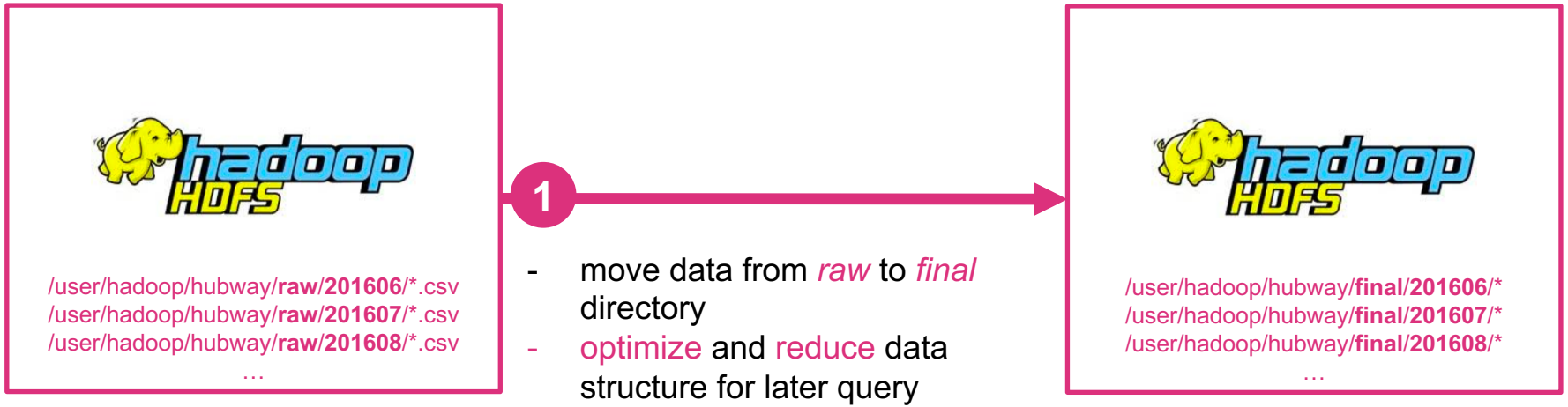
- **Gather data** from https://www.kaggle.com/acmeyer/hubway-data
- **Save raw data** (*CSV files*) to HDFS (partitioned by *YYYYMM*)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Calculate KPIs** and **Export** them to an **Excel File**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

Dataflow: 1. Get Hubway Bike Sharing Data



Dataflow: 2. Raw To Final Transfer



Dataflow: 3. Calculate And Export KPIs

Dataflow: 4. KPIs To Calculate

Use NYC Taxi Trip Record Data To Calculate Performance KPIs

Practical Exam

Goal

NYC.gov provides monthly exports of NYC taxi trip records:

- http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
- Latest Full Dumps:
  - *https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-06.csv*
  - *https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-05.csv*
  - *https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-04.csv*
  - *…*

```
1,2018-01-01 00:21:05,2018-01-01 00:24:23,1,.50,1,N,41,24,2,4.5,0.5,0.5,0,0,0.3,5.8
1,2018-01-01 00:44:55,2018-01-01 01:03:05,1,2.70,1,N,239,140,2,14,0.5,0.5,0,0,0.3,15.3
1,2018-01-01 00:08:26,2018-01-01 00:14:21,2,.80,1,N,262,141,1,6,0.5,0.5,1,0,0.3,8.3
1,2018-01-01 00:20:22,2018-01-01 00:52:51,1,10.20,1,N,140,257,2,33.5,0.5,0.5,0,0,0.3,34.8
1,2018-01-01 00:09:18,2018-01-01 00:27:06,2,2.50,1,N,246,239,1,12.5,0.5,0.5,2.75,0,0.3,16.55
1,2018-01-01 00:29:29,2018-01-01 00:32:48,3,.50,1,N,143,143,2,4.5,0.5,0.5,0,0,0.3,5.8
1,2018-01-01 00:38:08,2018-01-01 00:48:24,2,1.70,1,N,50,239,1,9,0.5,0.5,2.05,0,0.3,12.35
1,2018-01-01 00:49:29,2018-01-01 00:51:53,1,.70,1,N,239,238,1,4,0.5,0.5,1,0,0.3,6.3
[…]
```

yellow_tripdata_2018-06.csv

Goal

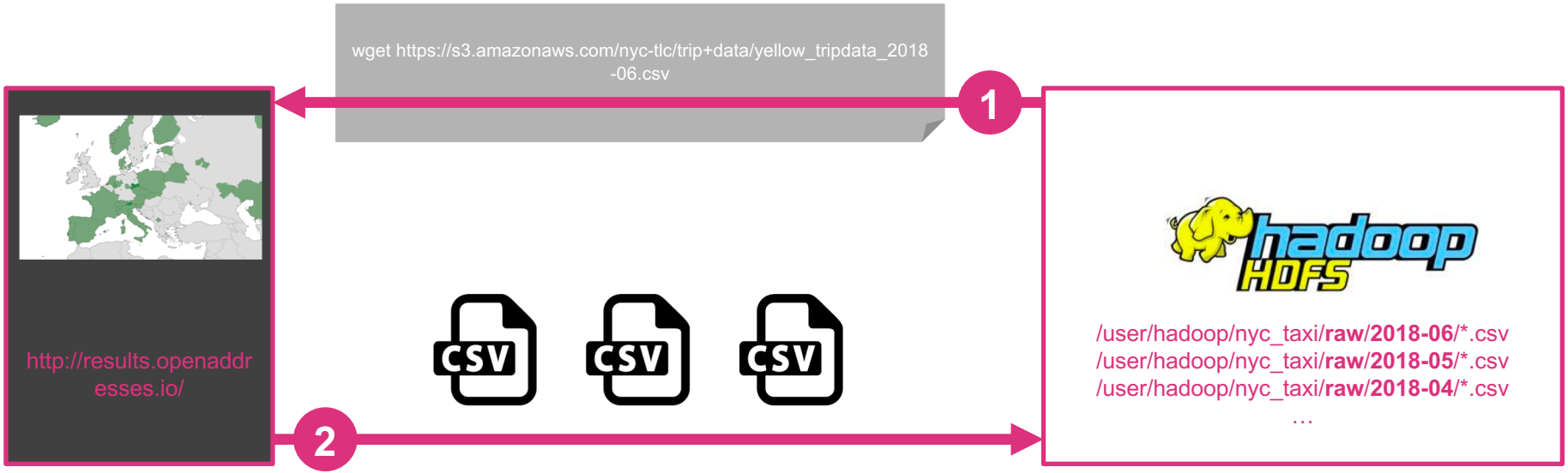
We want to make use of this data to calculate some KPIs

Workflow:

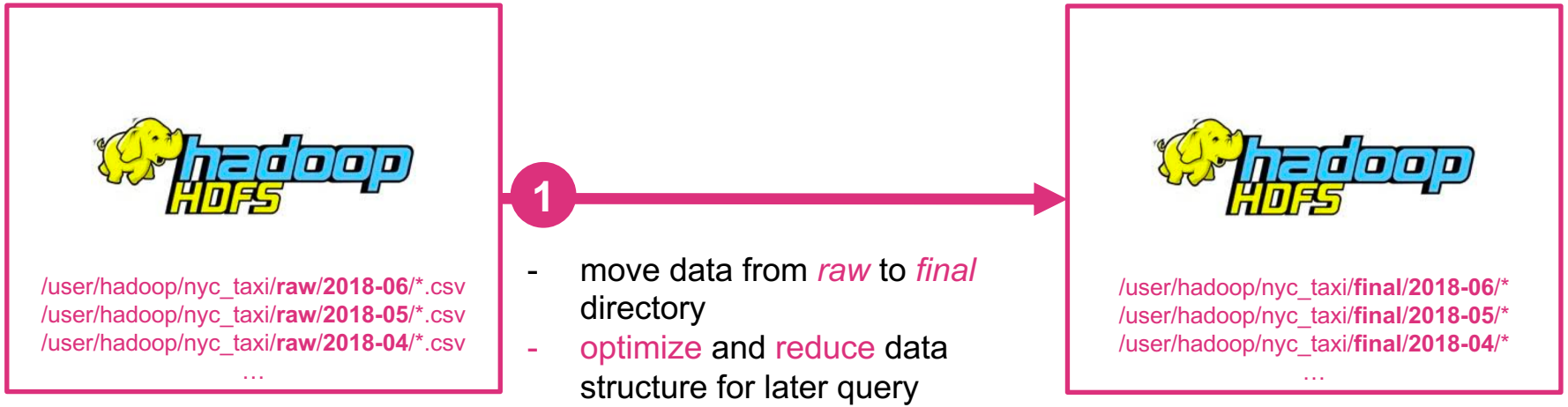
- **Gather data** from http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
- **Save raw data** (*CSV files*) to HDFS (partitioned by *YYYY-MM*)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Calculate KPIs** and **Export** them to an **Excel File**

- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

Dataflow: 1. Get TLC NYC Taxi Data



Dataflow: 2. Raw To Final Transfer



Dataflow: 3. Calculate And Export KPIs

Dataflow: 4. KPIs To Calculate

Create MTG Trading Card Database By API

Practical Exam

Goal

magicthegathering.io provides up-to-date information regarding all MTG trading cards available:

- https://docs.magicthegathering.io/
- E.g. https://api.magicthegathering.io/v1/cards/4711


```
{
  "card":{
    "name":"Manta Riders",
    "manaCost":"{U}",
    "cmc":1,
    "colors":[
      "Blue"
    ],
    "colorIdentity":[
      "U"
    ],
    "type":"Creature — Merfolk",
    "types":[
      "Creature"
    ],
    "subtypes":[
      "Merfolk"
    ],
    "rarity":"Common",
    "set":"TMP",
    "setName":"Tempest",
    "text":"{U}: Manta Riders gains flying until end of turn.",
    "flavor":"\"Water is firmament to the finned.\"\n—Oracle en-Vec",
    "artist":"Kaja Foglio",
    "power":"1",
    "toughness":"1",
    "layout":"normal",
    "multiverseid":4711,
    "imageUrl":"http://gatherer.wizards.com/Handlers/Image.ashx?multiverseid=4711&type=
card",
  […]
```

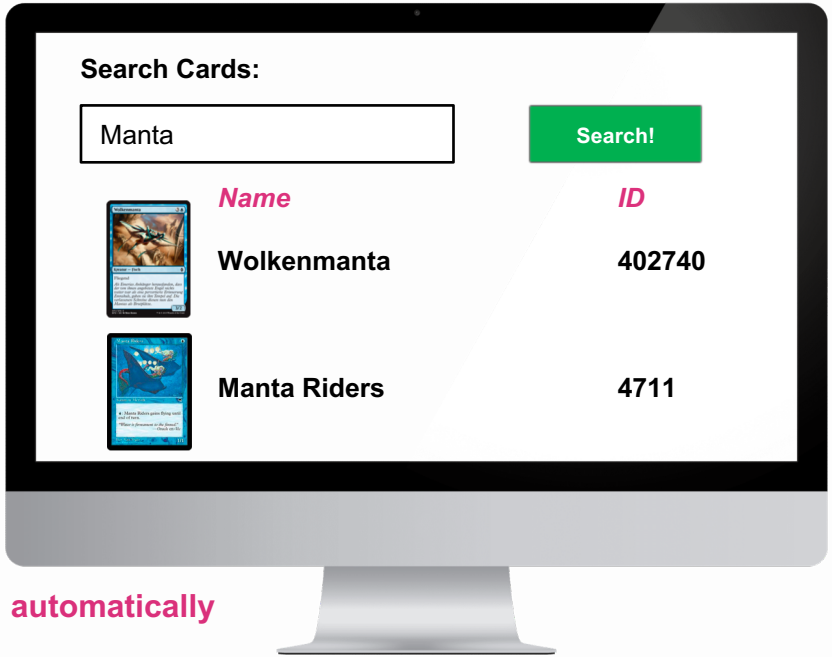
Goal

We want to make use of this data to build a searchable database of all MTG trading cards.

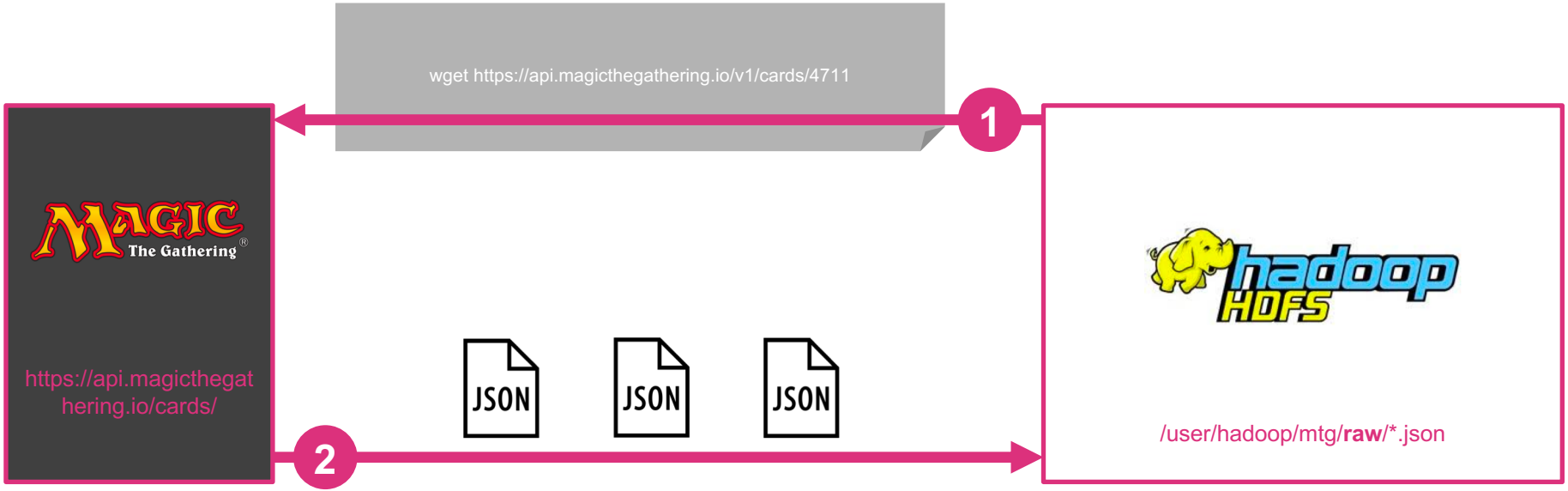
Workflow:

- **Gather data** from api.magicthegathering.io
- **Save raw data** (*JSON files*) to HDFS
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** MTG data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

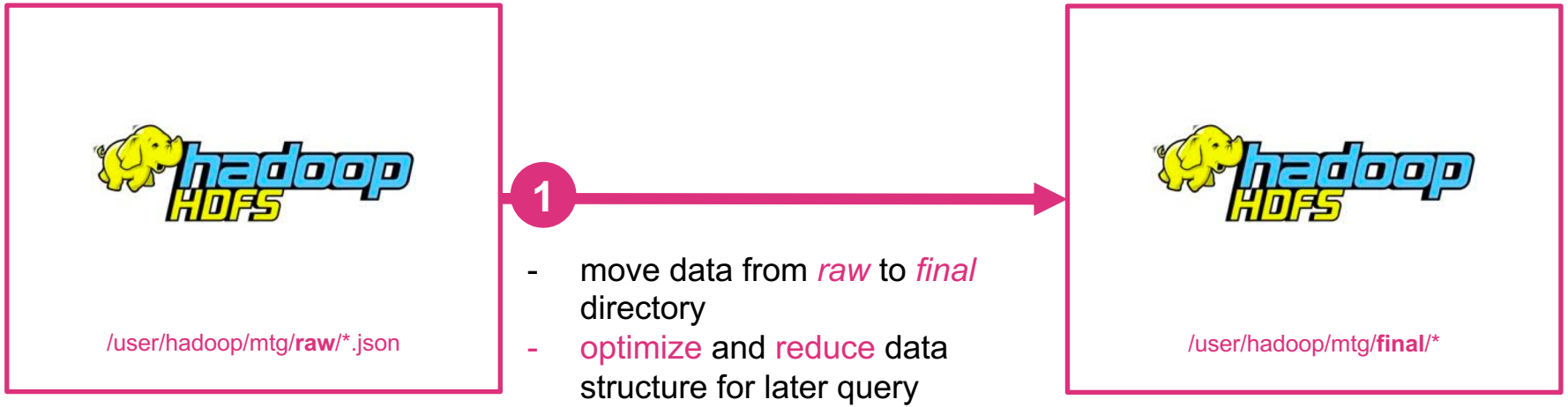
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



Dataflow: 1. Get MTG Data



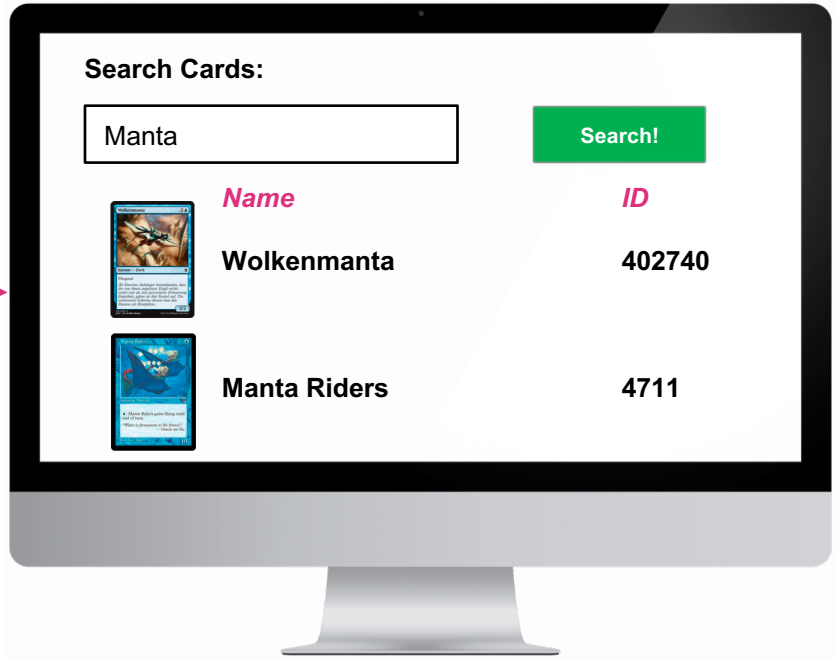
Dataflow: 2. Raw To Final Transfer



- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

Dataflow: 3. Enhance Data And Save Results

Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

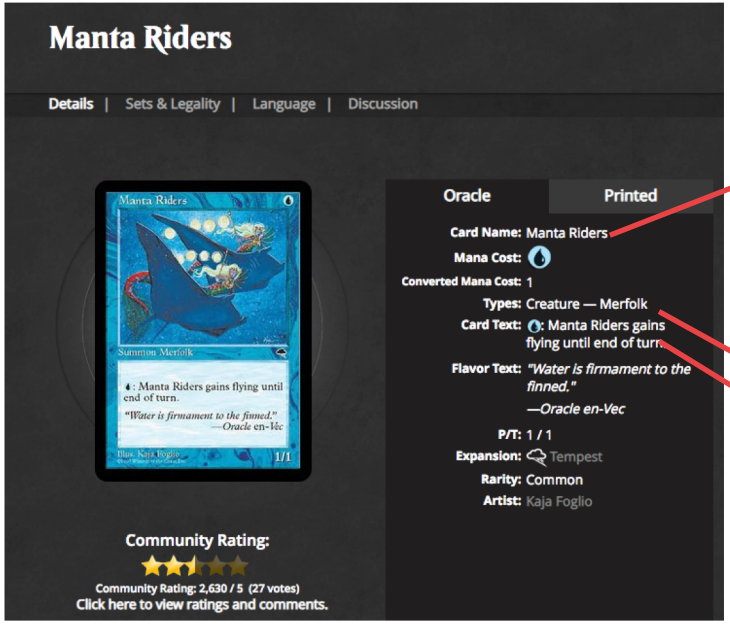
Create MTG Trading Card Database By Crawler

Practical Exam

Goal

magicthegathering.io provides up-to-date information regarding all MTG trading cards available:

- https://docs.magicthegathering.io/




```
<div class="smallGreyMono" style="margin-top: 5px;">
  <b class="ft"><b></b></b>
  <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_nameRow" class="row">
    <div class="label">
      Card Name:</div>
    <div class="value">
      Manta Riders</div>
  </div>
  <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_manaRow" class="row manaRow">
    <div class="label" style="line-height: 25px;">
      Mana Cost:</div>
    <div class="value">
      <img src="/Handlers/Image.ashx?size=medium&amp;name=U&amp;type=symbol" alt="Blue" align="absbottom" /></div>
  </div>
  <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_cmcRow" class="row">
    <div class="label" style="font-size: .7em;">
      Converted Mana Cost:</div>
    <div class="value">
      1</div>
  </div>
  <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_typeRow" class="row">
    <div class="label">
      Types:</div>
    <div class="value">
      Creature — Merfolk</div>
  </div>
  <div id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_textRow" class="row">
    <div class="label">
      Card Text:</div>
    <div class="value">
      <div class="cardtextbox" style="padding-left:10px;"><img src="/Handlers/Image.ashx?size=small&amp;name=U&amp;type=sy
mbol" alt="Blue" align="absbottom" />: Manta Riders gains flying until end of turn.</div></div>
  </div>
```

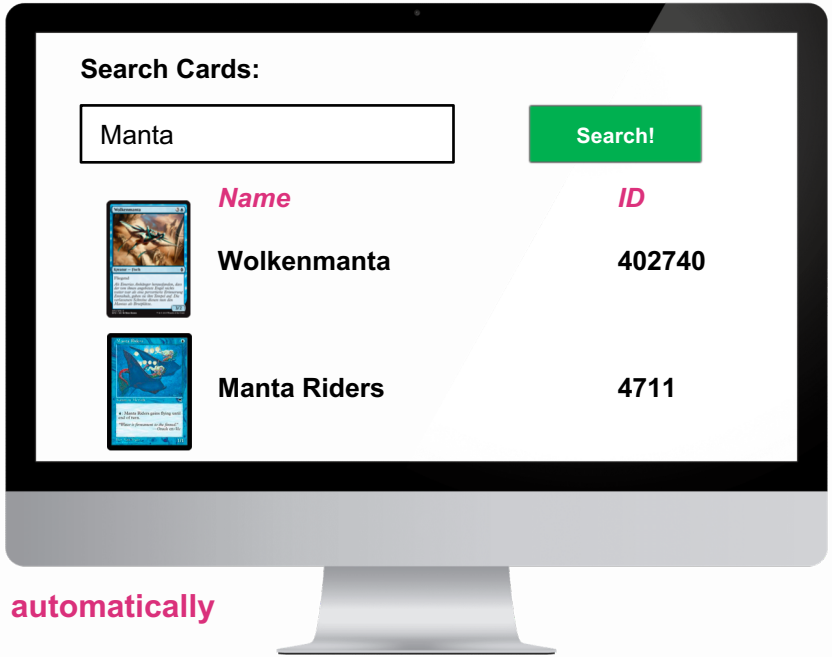
http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=4711

Goal

We want to make use of this data to build a searchable database of all MTG trading cards.

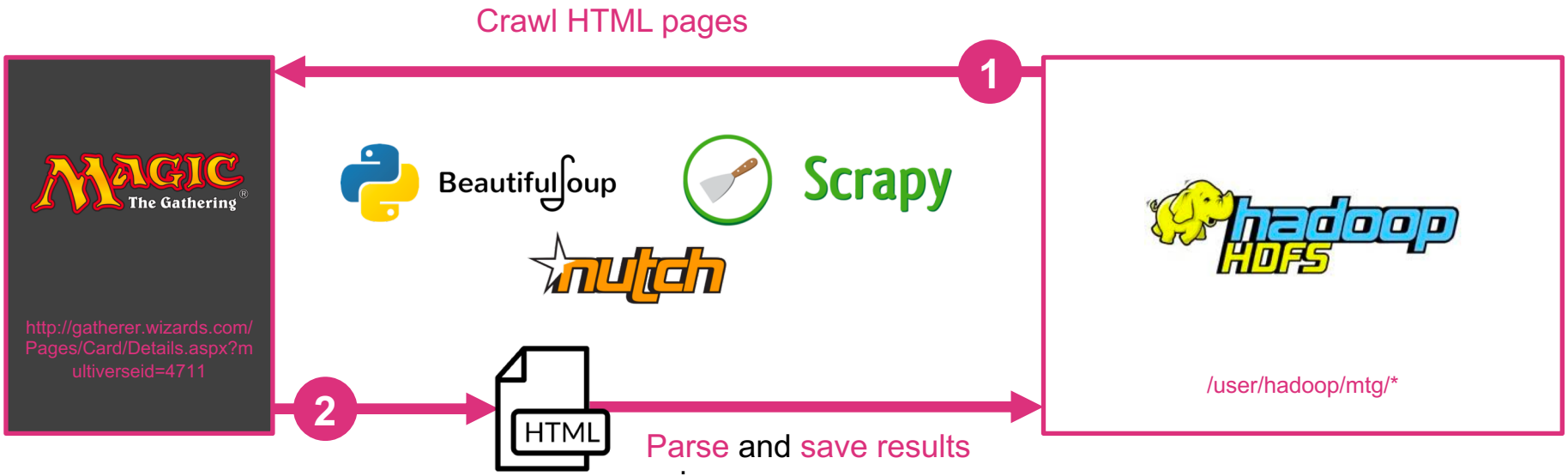
Workflow:

- **Crawl data** from gatherer.wizards.com
- **Parse** required **information** and save them to **HDFS** (queryable through Hive)
- **Export** MTG data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**



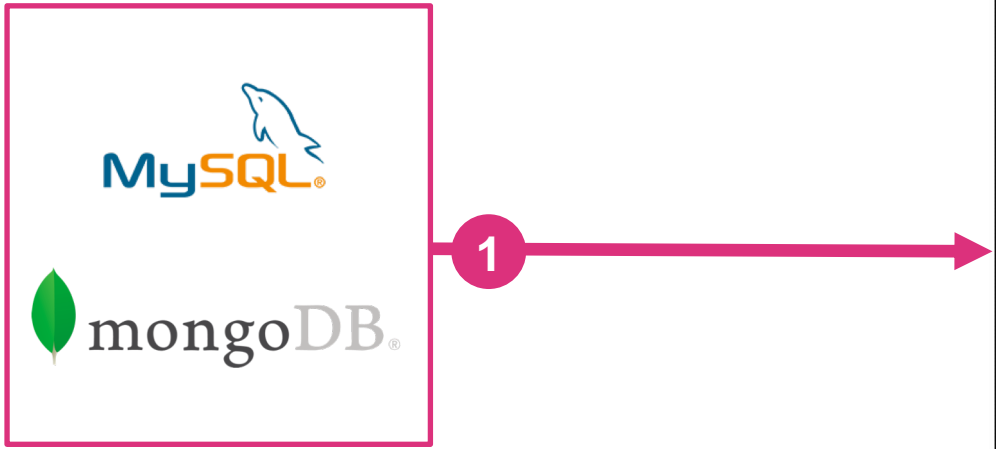
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

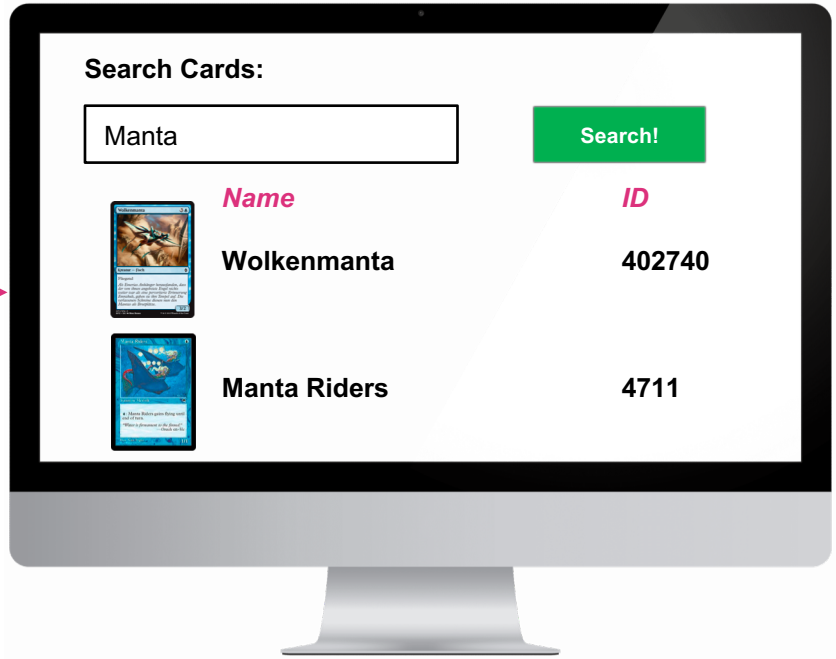
Dataflow: 1. Get MTG Data



Dataflow: 3. Enhance Data And Save Results

Dataflow: 4. Provide Simple Web Interface





- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (card name, text or artist)
  - **display search results**

Use GeoLite2 To Create A Searchable IP and GeoLocation Database

Practical Exam

Goal

Maxmind.com provides regulary exports of worldwide IP and Geolocation data:

- https://dev.maxmind.com/geoip/geoip2/geolite2/

```
curl -s http://whatismyip.akamai.com/
88.130.59.75
```

1

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname_id,is_anonymous_proxy,is_satellite_provider,postal_code,latitude,longitude,accuracy_radius
88.130.59.0/24,2939623,2921044,,0,0,85221,48.2600,11.4340,50
[…]
```

GeoLite2-City-Blocks-IPv4.csv

2

```
geoname_id,locale_code,continent_code,continent_name,country_iso_code,country_name,subdivision_1_iso_code,subdivision_1_name,subdivision_2_iso_code,subdivision_2_name,city_name,metro_code,time_zone,is_in_european_union
3209535,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Höhenkirchen-Siegertsbrunn,,Europe/Berlin,1
2939623,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Dachau,,Europe/Berlin,1
3207410,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Rödental,,Europe/Berlin,1
3207412,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Röslau,,Europe/Berlin,1
3208324,de,EU,Europa,DE,Deutschland,BY,Bayern,,,Asbach-Bäumenheim,,Europe/Berlin,1
[…]
```

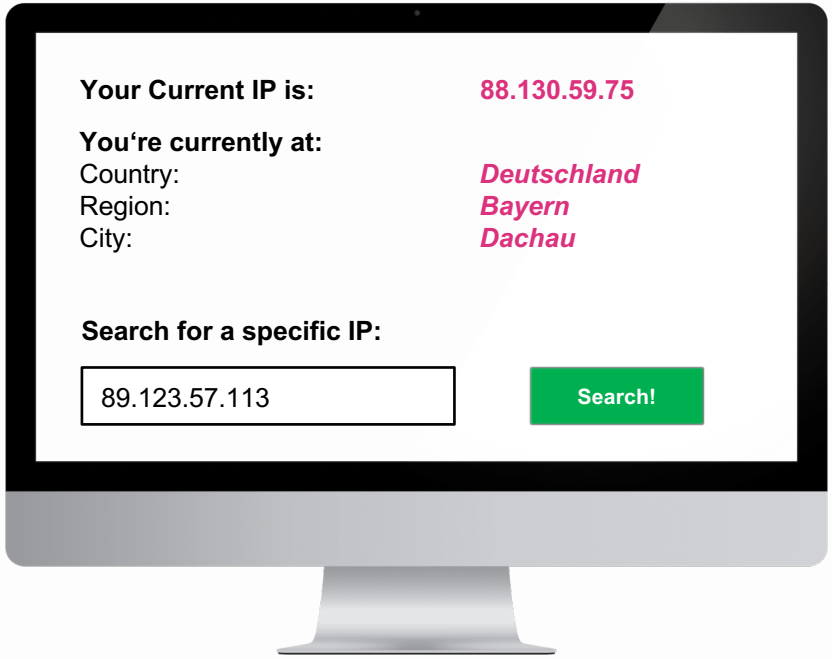
GeoLite2-City-Locations-[XX].csv

Goal

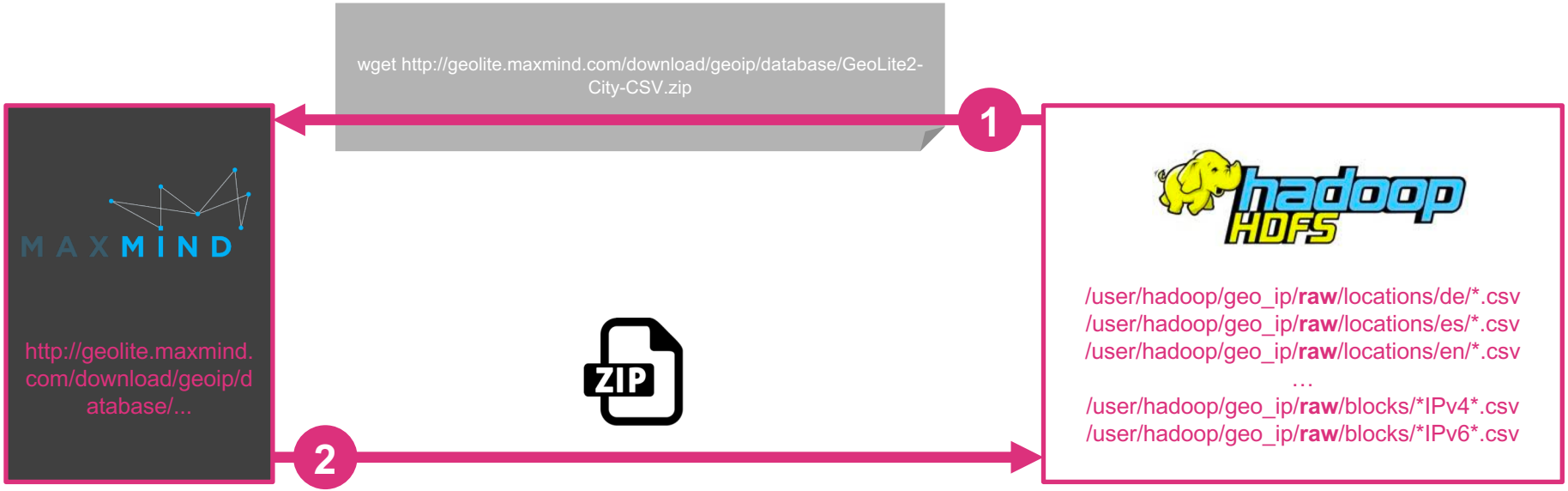
We want to make use of this data to build a real time IP-Geolocation resolution as well as a searchable database for Ips and related Geolocations.

Workflow:

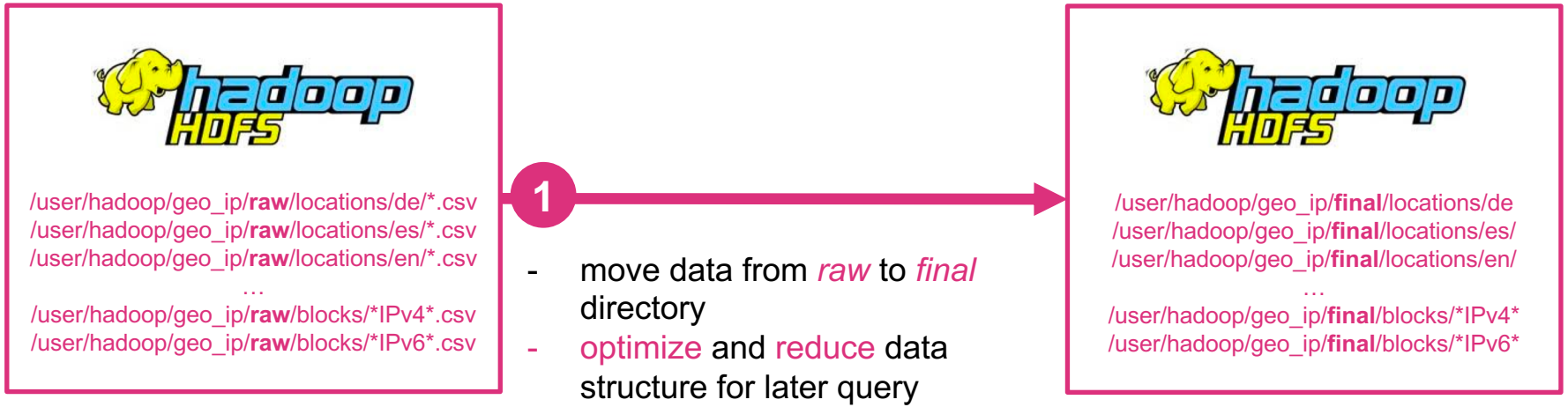
- **Gather data** from maxmind.com
- **Save raw data** (*CSV files*) to HDFS (partitioned by country code, e.g. *de*, *es*, *en*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** Geolite2 data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user's IP address, **lookup** and **show Geolocation**
  - process user input (IP…) and check against end-user database
  - Display result Geolocation
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



Dataflow: 1. Get Geolite2 Data



Dataflow: 2. Raw To Final Transfer

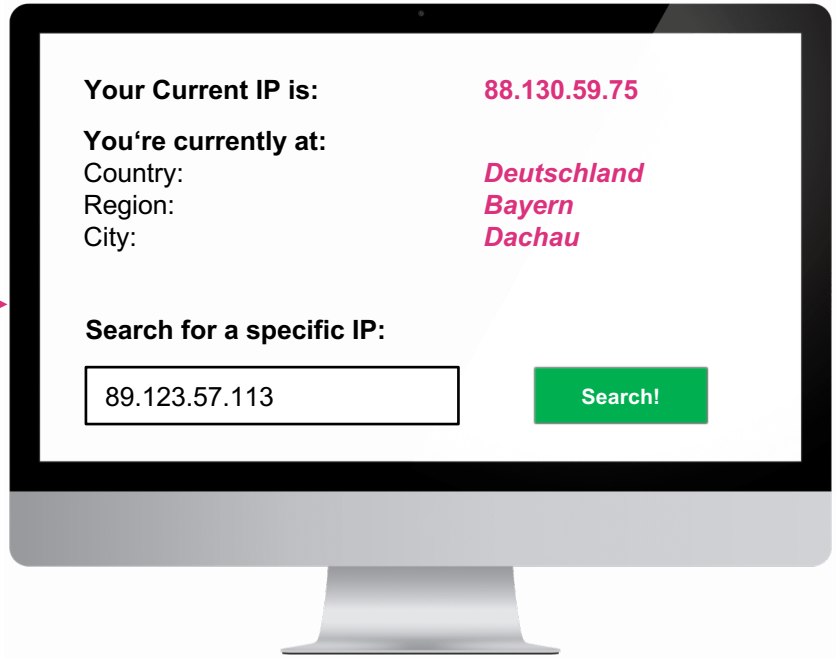


- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

Dataflow: 3. Enhance Data And Save Results

- enhance data (e.g. for later querying)
- use *Hive*, *Python*, *Spark* or *PySpark*
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - **determine** a user's IP address, **lookup** and **show** Geolocation
  - process user input (IP…) and check against end-user database
  - Display result Geolocation

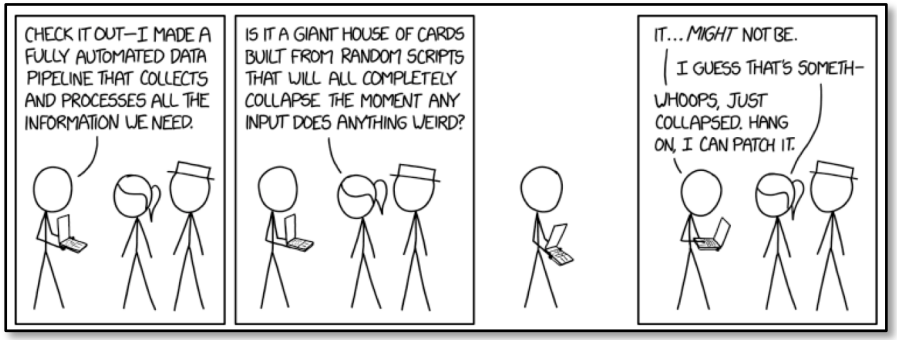
Use XKCD API To Build A Searchable Database of XKCD Comics

Practical Exam

Goal

OpenCellID.com provides regulary exports of worldwide cell data:

- https://xkcd.com/
- JSON API: https://xkcd.com/2054/info.0.json



https://xkcd.com/2054/


```
{
        "month": "10",
        "num": 2054,
        "link": "",
        "year": "2018",
        "news": "",
        "safe_title": "Data Pipeline",
        "transcript": "",
        "alt": "\"Is the pipeline literally running from your laptop?\" \"Don't b
e silly, my laptop disconnects far too often to host a service we rel
y on. It's running on my phone.\"", "img": "https://imgs.xkcd.com/comi
cs/data_pipeline.png",
        "title": "Data Pipeline",
        "day": "3"
}
```

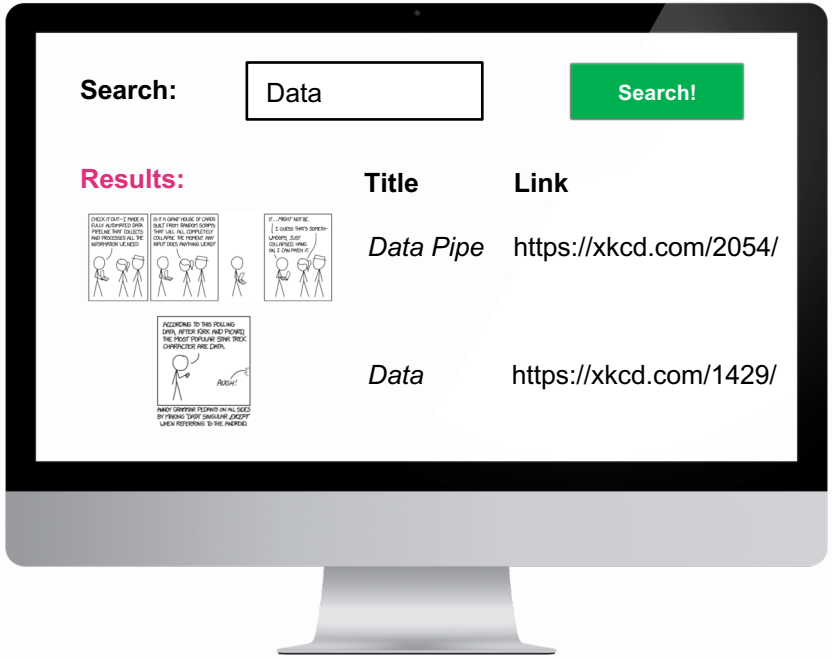

2054.json

Goal

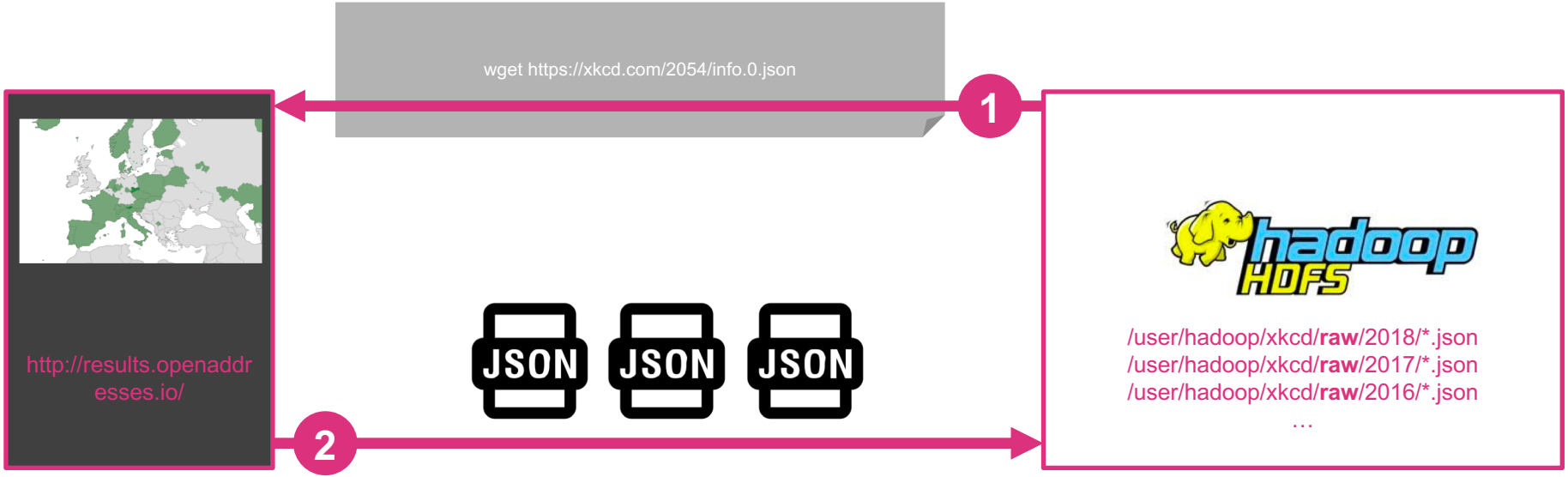
We want to make use of this data to build a searchable database for XKCD comics.

Workflow:

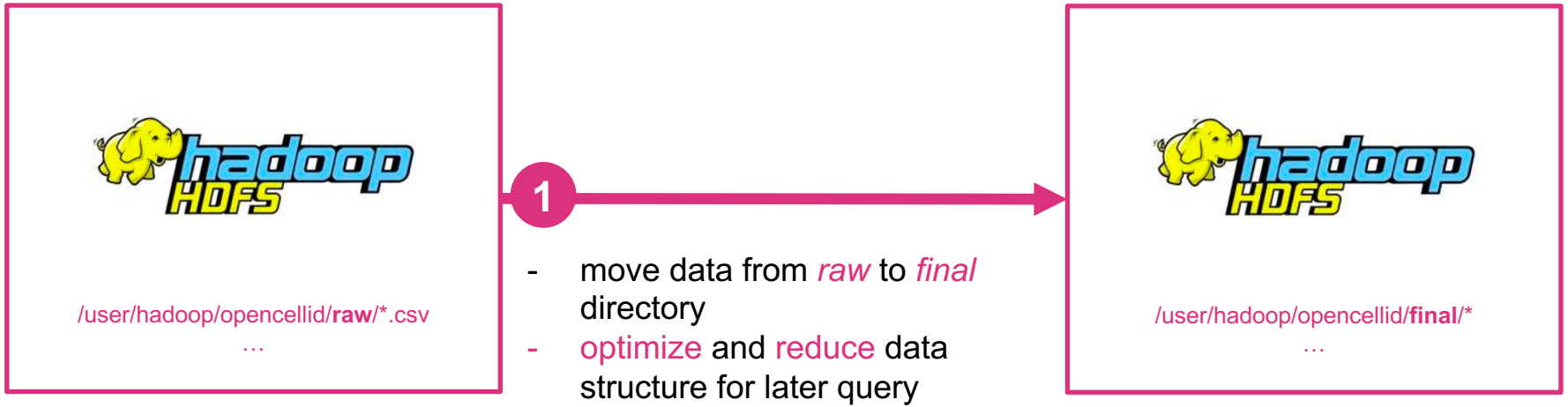
- **Gather data** from xkcd.com
- **Save raw data** (*JSON files*) to HDFS (partitioned by year, e.g. *2018*, *2017*, *2016*…)
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Export** xkcd data **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (search phrase…)
  - checks against xkcd data in end-user database
  - Display result (comics containing search phrase)
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**



Dataflow: 1. Get XKCD Data



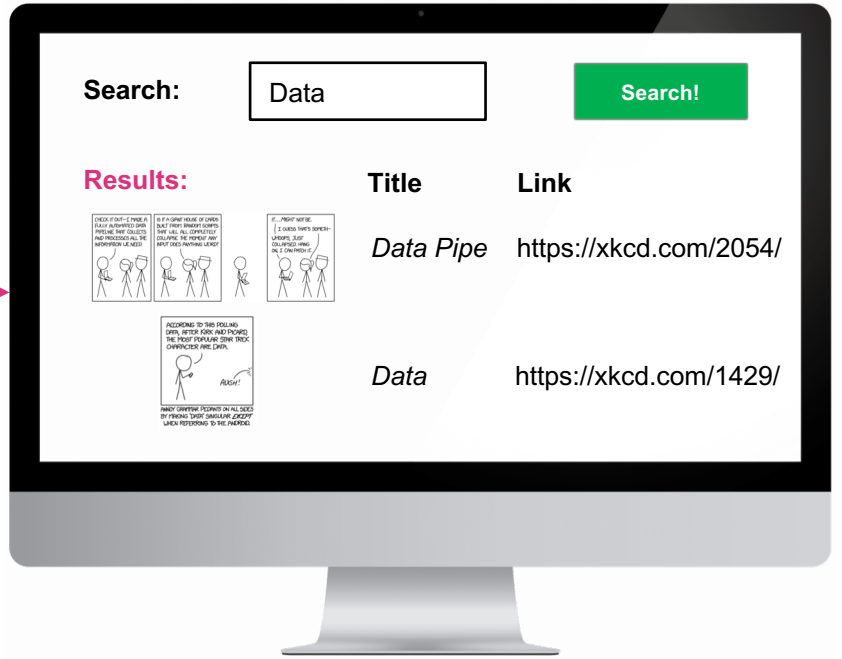
Dataflow: 2. Raw To Final Transfer



- move data from *raw* to *final* directory
- optimize and reduce data structure for later query purposes if necessary
- remove duplicates if necessary
- …

Dataflow: 3. Enhance Data And Save Results

Dataflow: 4. Provide Simple Web Interface



- Provide a simple **HTML Frontend** which is able to:
  - read from end-user database
  - process user input (search phrase…)
  - checks against xkcd data in end-user database
  - Display result (comics containing search phrase)

Use Spotify Audio Features To Categorize Music

Practical Exam

Goal

Spotify provides an API for basic track information:

- **`ID`**, e.g. *2lEcSduKEXEK5KJ9hJzlCz*
- **`name`**, e.g. *Gloana Bauer (Teenage Dirtbag)*
- **`artist`**, e.g. *D‘ Hundskrippln*
- *…*

as well as related audio features:

- **`energy`**, e.g. *0.454*
  Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.
- **`speechiness`**, e.g. *0.0388*
  Speechiness detects the presence of spoken words in a track. The more exclu sively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
- **`loudness`**, e.g. *-8.758*
  The overall loudness of a track in decibels (dB). Loudness values are averaged.
- **`tempo`**, e.g. *97.532*
  The overall estimated tempo of a track in beats per minute (BPM)
- **`acousticness`**, *e.g. 0.268*
  A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- …

https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

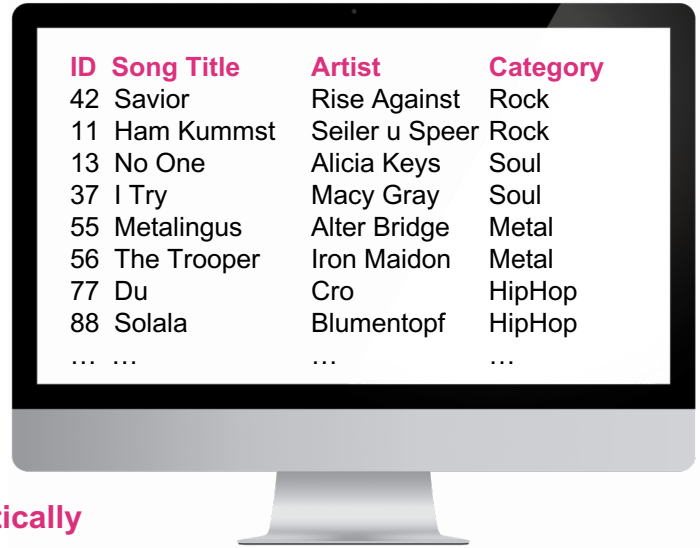
Goal

We want to make use of those audio features to automatically assign each track to a certain category:

Metal	***Classic***	***Rock***	***Vocal***
Electro	***Podcast***	***Soul***	***HipHop***

Workflow:

- **Query data** from Spotify API
- **Save raw data** (*JSON* files) to HDFS
- **Optimize**, **reduce** and **clean raw data** and save it to **final** directory on HDFS
- **Calculate categories** (*Metal, Classic, Rock, …*)
- **Join track information** and **audio festures** and **save** everything **to end-user database** (e.g. MySQL, MongoDB…)
- Provide a simple HTML Frontend which reads from end-user database and displays result
- The whole data workflow **must be implemented** within an ETL **workflow tool** (e.g. Pentaho Data Integration or Airflow) and **run automatically**

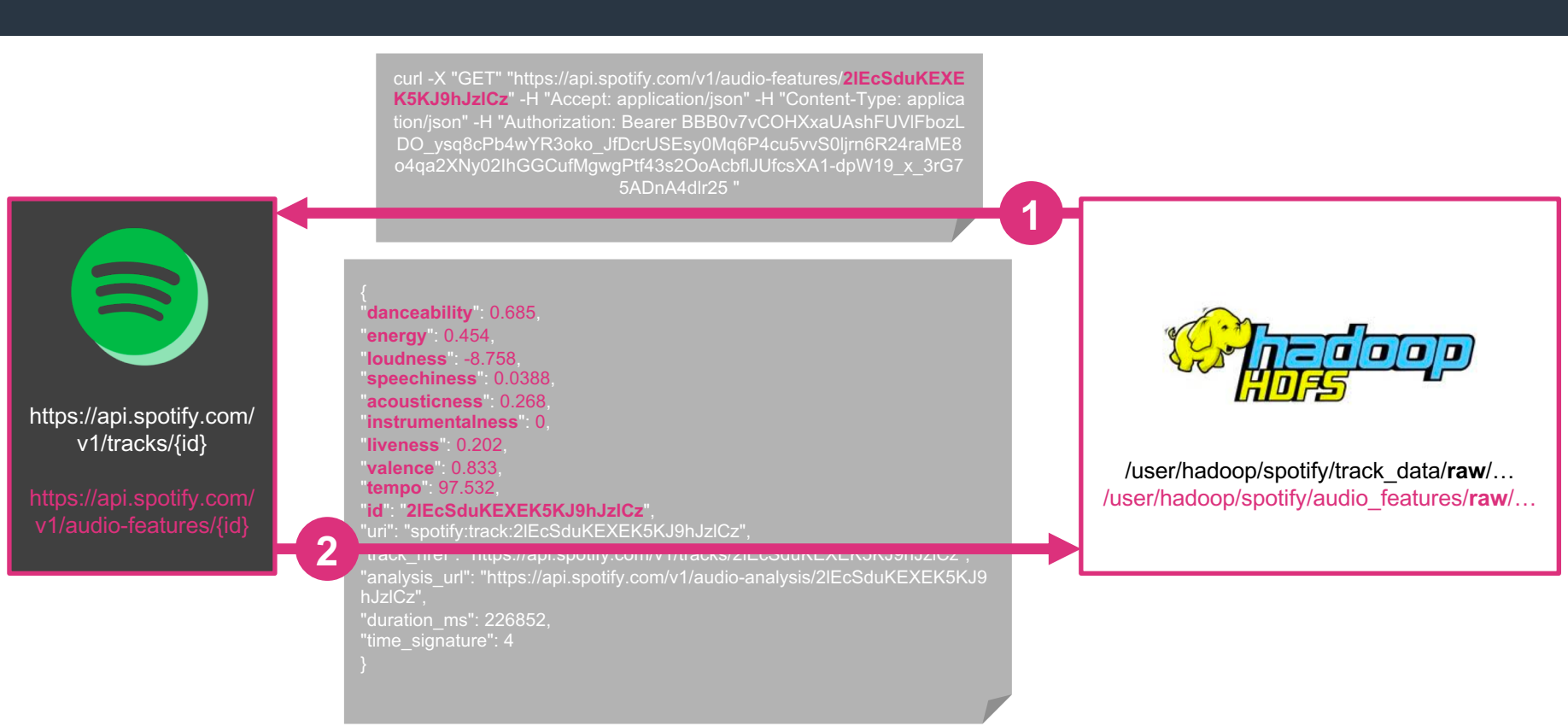


ID	Song Title	Artist	Category
42	Savior	Rise Against	Rock
11	Ham Kummst	Seiler u Speer	Rock
13	No One	Alicia Keys	Soul
37	I Try	Macy Gray	Soul
55	Metalingus	Alter Bridge	Metal
56	The Trooper	Iron Maidon	Metal
77	Du	Cro	HipHop
88	Solala	Blumentopf	HipHop
…	…	…	…

Dataflow: 1. Get Track Information

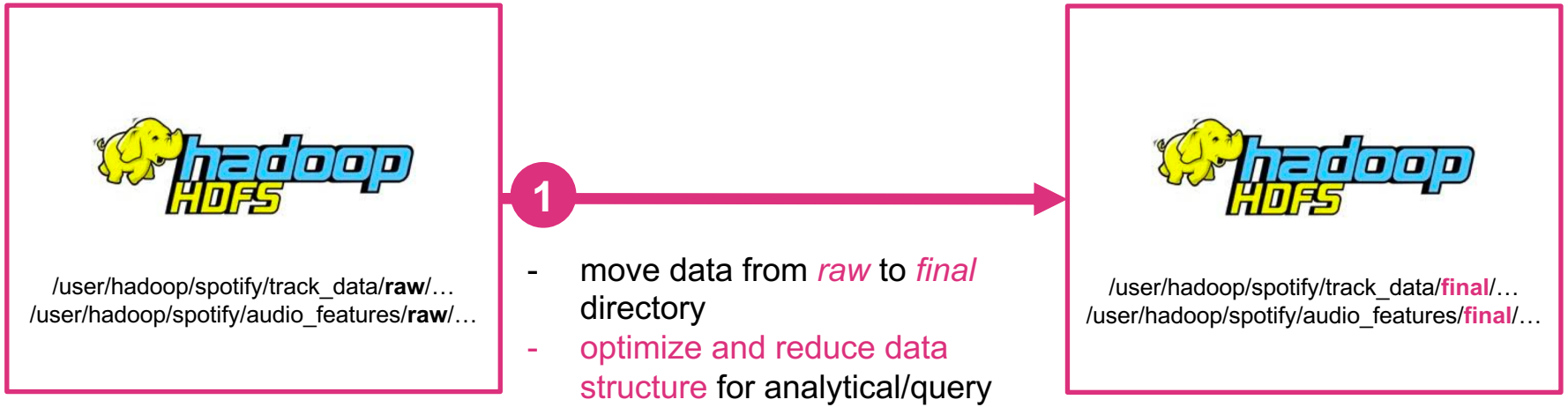
https://developer.spotify.com/console/get-track/

Dataflow: 2. Get Track Audio Features



https://developer.spotify.com/console/get-audio-features-track/

Dataflow: 3. Raw To Final Transfer



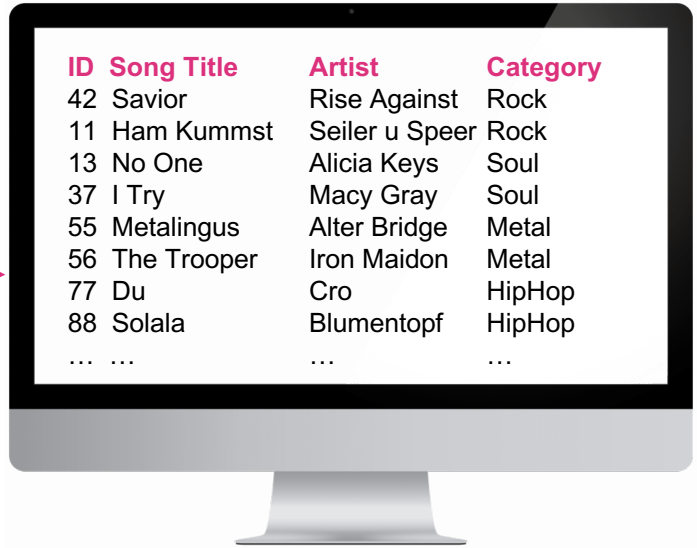
Dataflow: 4. Run Analysis and Save Results

- calculate categories for each track, using *Hive*, *Python*, *Spark* or *PySpark*
- join *track* and *audio feature* data
- save everything to a end-user database (e.g. *MySQL*, *MongoDB*)

Dataflow: 5. Provide Simple Web Interface

1

- Create a simple Website which reads and displays data from end-user database



ID	Song Title	Artist	Category
42	Savior	Rise Against	Rock
11	Ham Kummst	Seiler u Speer	Rock
13	No One	Alicia Keys	Soul
37	I Try	Macy Gray	Soul
55	Metalingus	Alter Bridge	Metal
56	The Trooper	Iron Maidon	Metal
77	Du	Cro	HipHop
88	Solala	Blumentopf	HipHop
…	…	…	…