

Big Data — рабочие процессы и автоматизация, пакетная и потоковая обработка

Batch vs Stream Processing

Batch Processing

Stream Processing

Тип данных:

large, historic

volatile, live, stream

Время выполнения:

minutes, hours, days

real-time/near-real-time

Повторное выполнение: possible

„impossible“

Batch Processing – Example Data Flow

Source Systems:

e.g. CRM, ERP, Webserver
Logfiles, ...



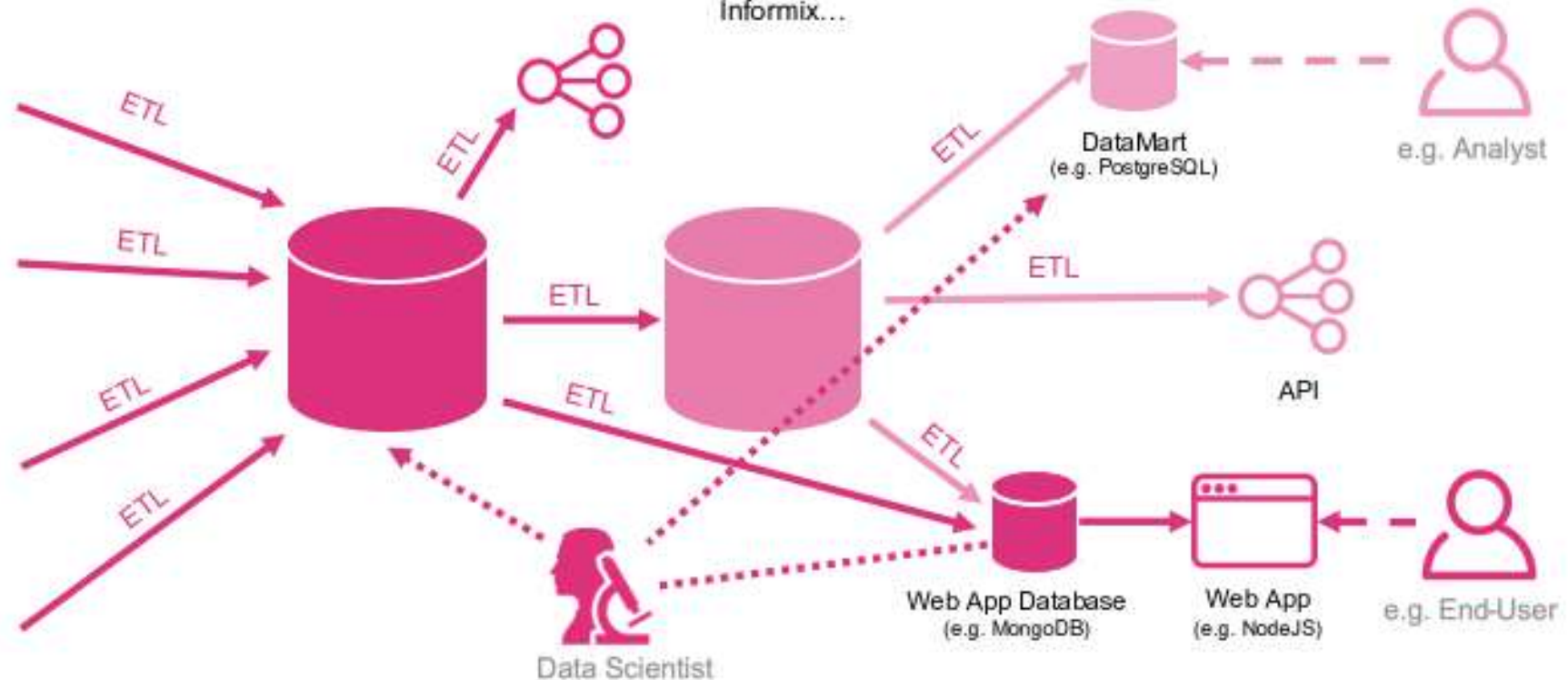
Data Lake:

e.g. Hadoop HDFS

Data Warehouse:

e.g. PostgreSQL, Oracle,
MS SQL Server, DB2,
Informix...

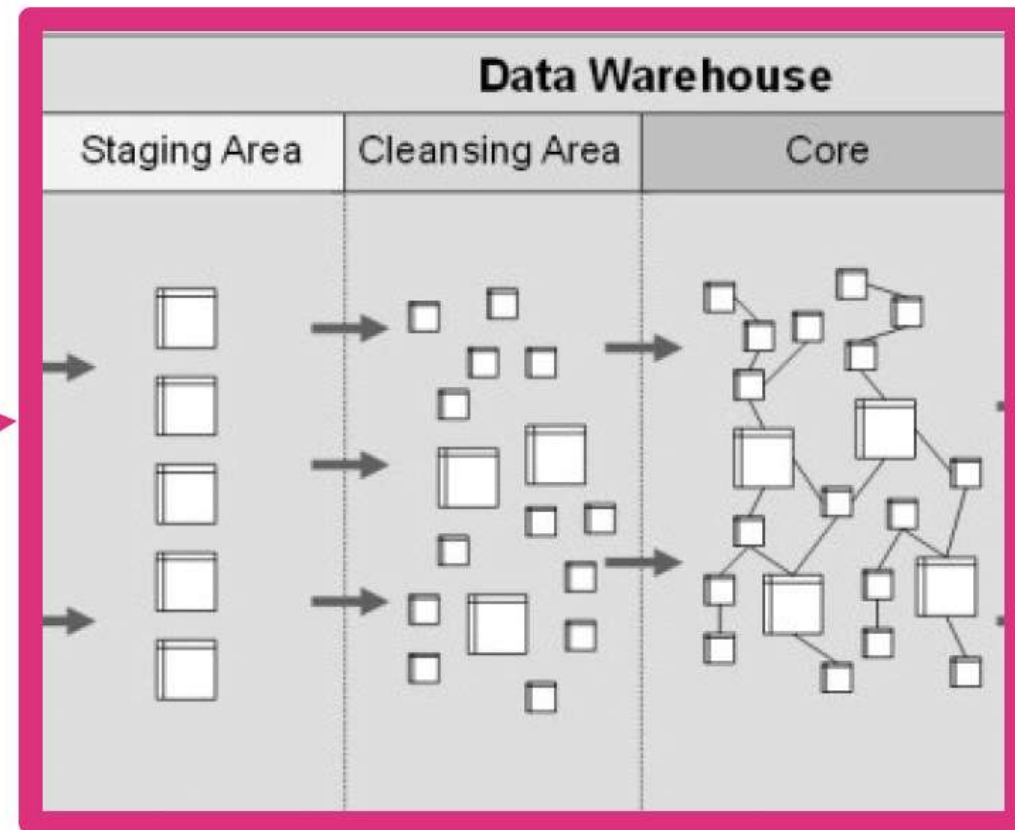
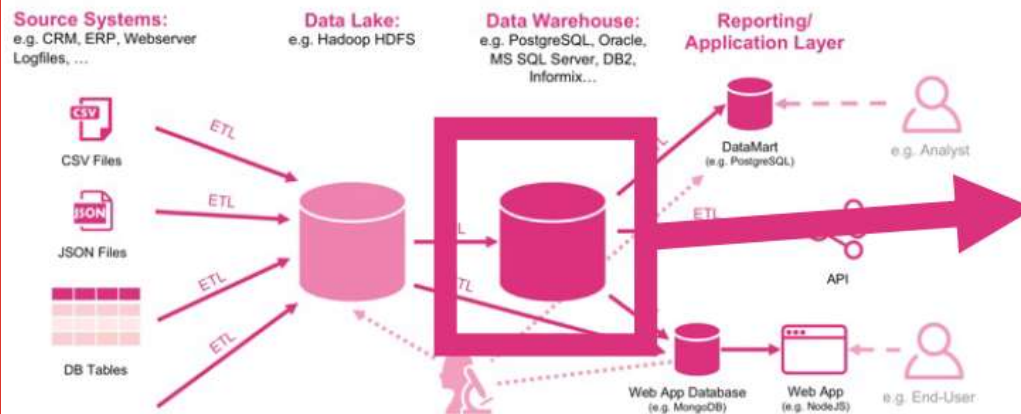
Reporting/ Application Layer



ETL - **Extract, Transform, Load**(извлечение, преобразование, загрузка) — это базовый шаблон обработки данных, широко известный в хранилищах данных.

Все дело в извлечении данных из источника, преобразовании данных (например, путем применения бизнес-правил или изменения структур) и записи/загрузки всего в целевой объект (например, **Hadoop HDFS**, **Hive**, реляционная база данных, хранилище данных, витрина данных и т. д.)

Batch Processing – пример Data Flow



Batch Processing— диссоциация хранилища данных



- Система Big Data может быть **частью** или **источником** хранилища данных, например:

- **Data Lake.**
- **Enterprise Data Hub.**

Data Warehouse

- обрабатывает в основном структурированные данные
- подходит для небольших объемов данных;
- фокусируется на аналитических и отчетных задачах;
- 100% точность .

Big Data

- обрабатывает данные в любой структуре;
- служит широкому кругу задач, связанных с данными (например, аналитика, наука о данных, приложения, основанные на данных, ...);
- не 100% точность.

Distributed Batch Processing

Distributed Storage



redis



mongoDB®

Distributed Processing

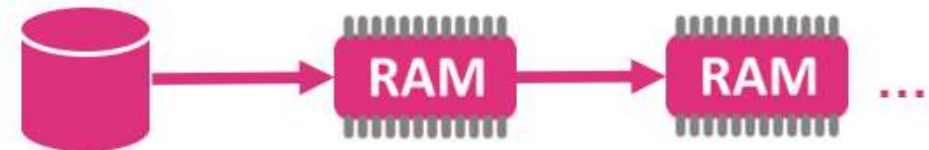


Batch Processing – MapReduce vs Spark

Hadoop MapReduce



Spark



- Хорошая **надежность**
- Плохая **производительность**

- Плохая **надежность**
- Хорошая **производительность**

Spark vs Flink

reliability > latency



- **структура пакетной обработки**, которая эмулирует потоковую обработку
- **потоковая обработка** = выполнение микропакетов
- режет поток данных на микропакеты и обрабатывает каждый пакет
- Задержка выполнения: (несколько) секунд
- Степень внедрения: высокая, множество библиотек, огромное сообщество и база разработчиков

latency > reliability



- **структура потоковой обработки**, которая эмулирует пакетную обработку
- **пакетная обработка** = обработка ограниченного потока
- обрабатывает каждое событие, когда оно приходит
- Задержка выполнения: миллисекунды-секунды
- Степень внедрения: средняя, ограниченные библиотеки, среднее сообщество и база разработчиков

Message Broker/Queues

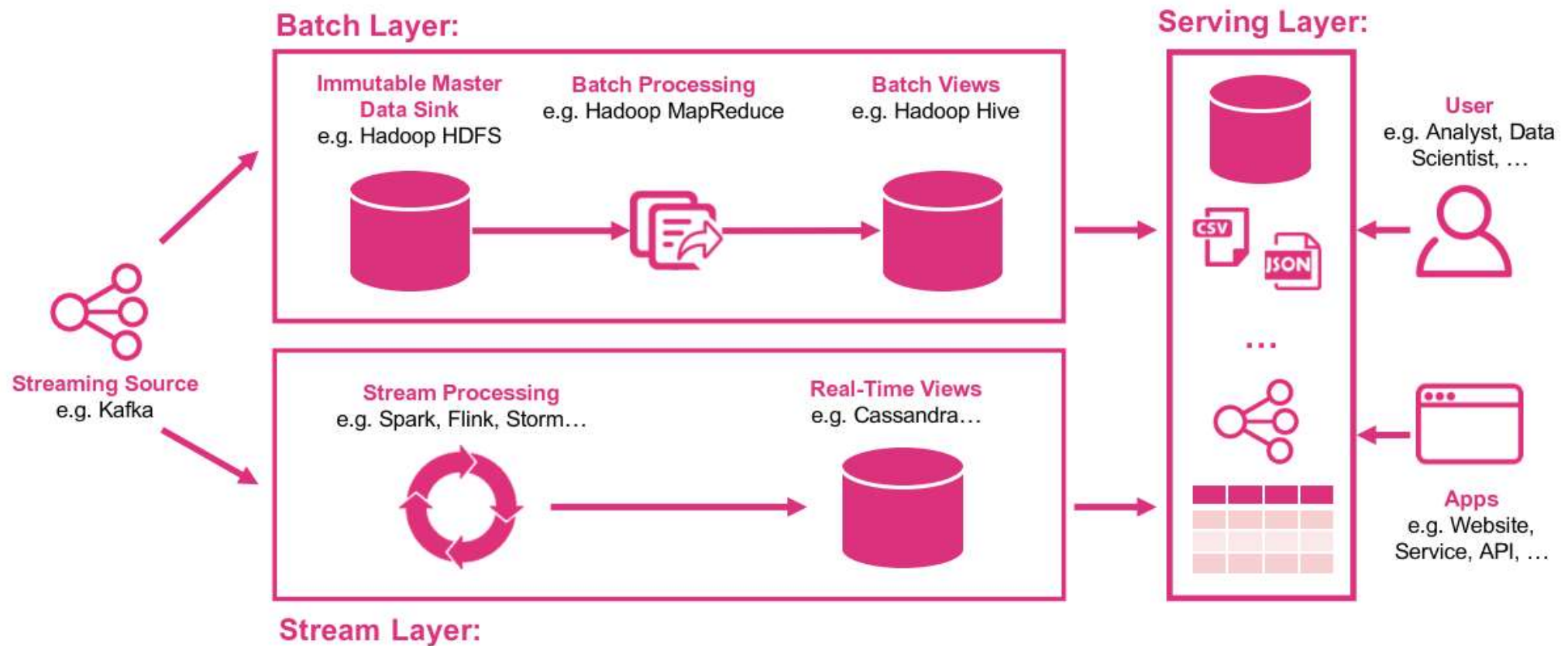
Зачем:

- разделяет отправителя и получателя
- поддерживает очереди разного уровня
- сохраняются временные сообщения
- Уведомляет подписчиков о новых сообщениях
- Микросервисы, IoT.

Примеры:

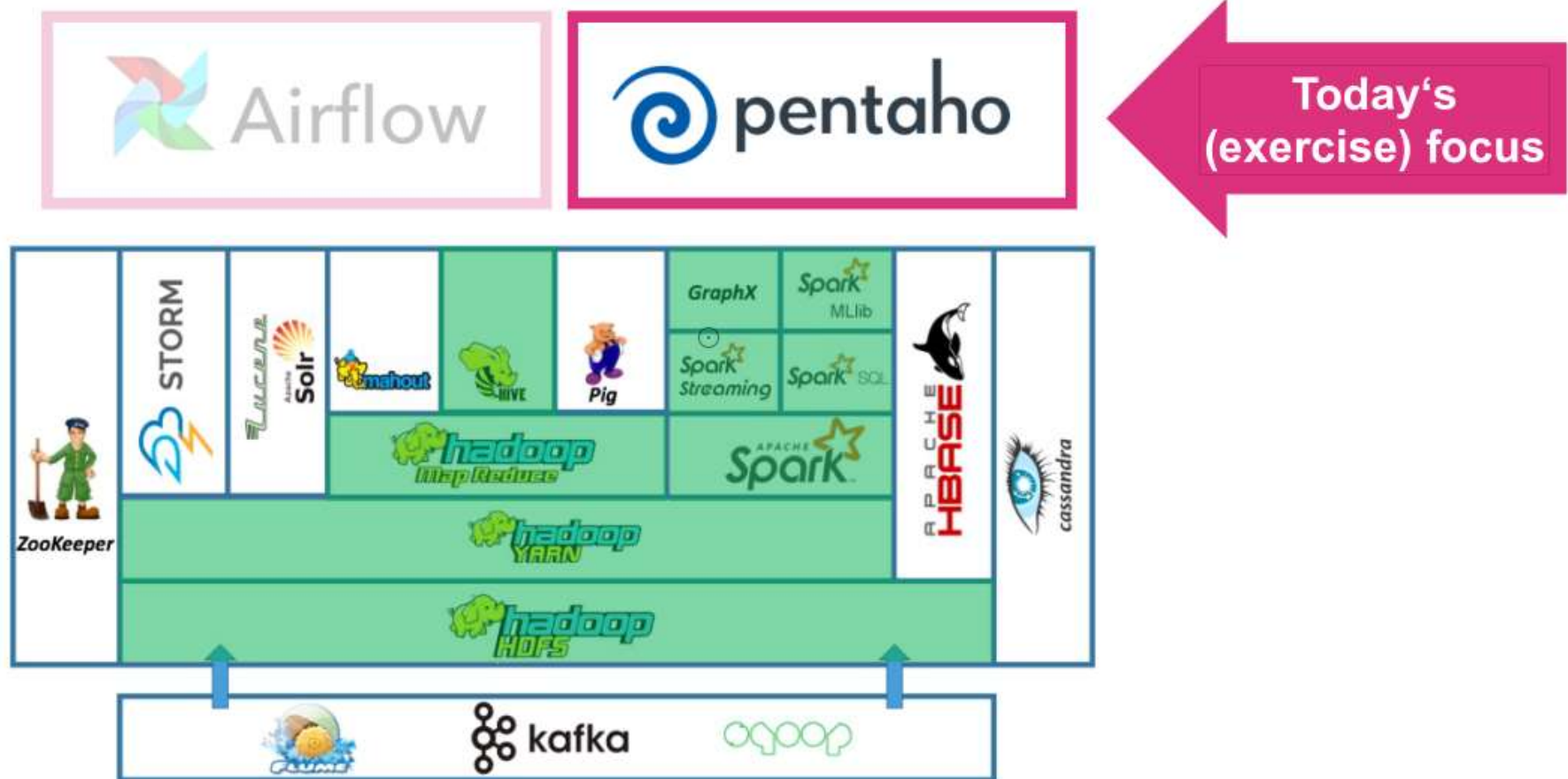
- RabbitMQ,
- ZeroMQ,
- Kafka,
- ActiveMQ,
- Kestrel.

Batch&Stream Processing – Lambda Architecture



Подготовка к упражнениям Pentaho Data Integration

The Hadoop Ecosystem



Download And Install PDI

1. Download Pentaho Data Integration (9.3)

<https://sourceforge.net/projects/pentaho/>

<https://disk.yandex.ru/d/iVAdHZAPliBbkQ>

2. Extract:

```
unzip pdi-ce-9.3.0.0-428.zip
```

Start PDI

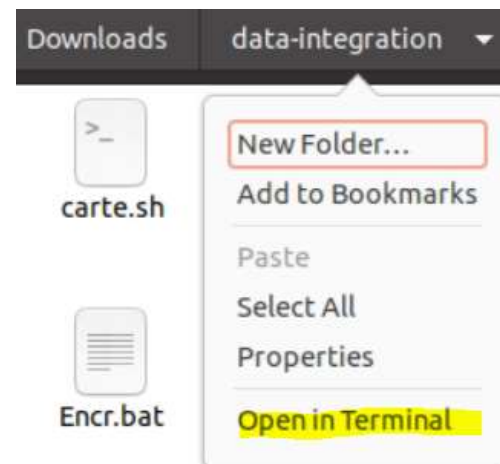
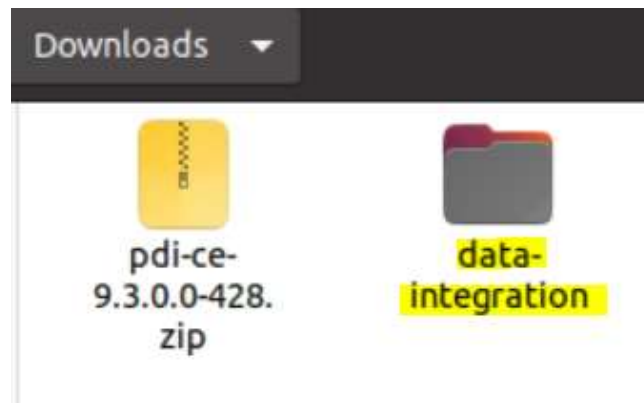
3. Start Pentaho Data Integration

LINUX

`/your/pentaho/directory/spoon.sh`

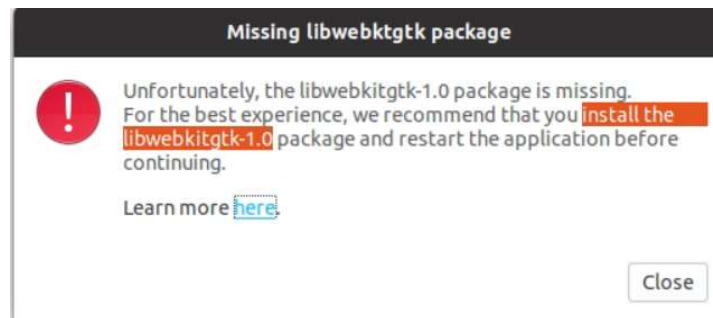
WINDOWS

`/your/pentaho/directory/spoon.bat`



3. Start Pentaho Data Integration

```
lemp001@u20-16:~/Downloads/data-integration$ sudo sh spoon.sh
```



```
sudo nano /etc/apt/sources.list
```

Add this entry to the file and save:

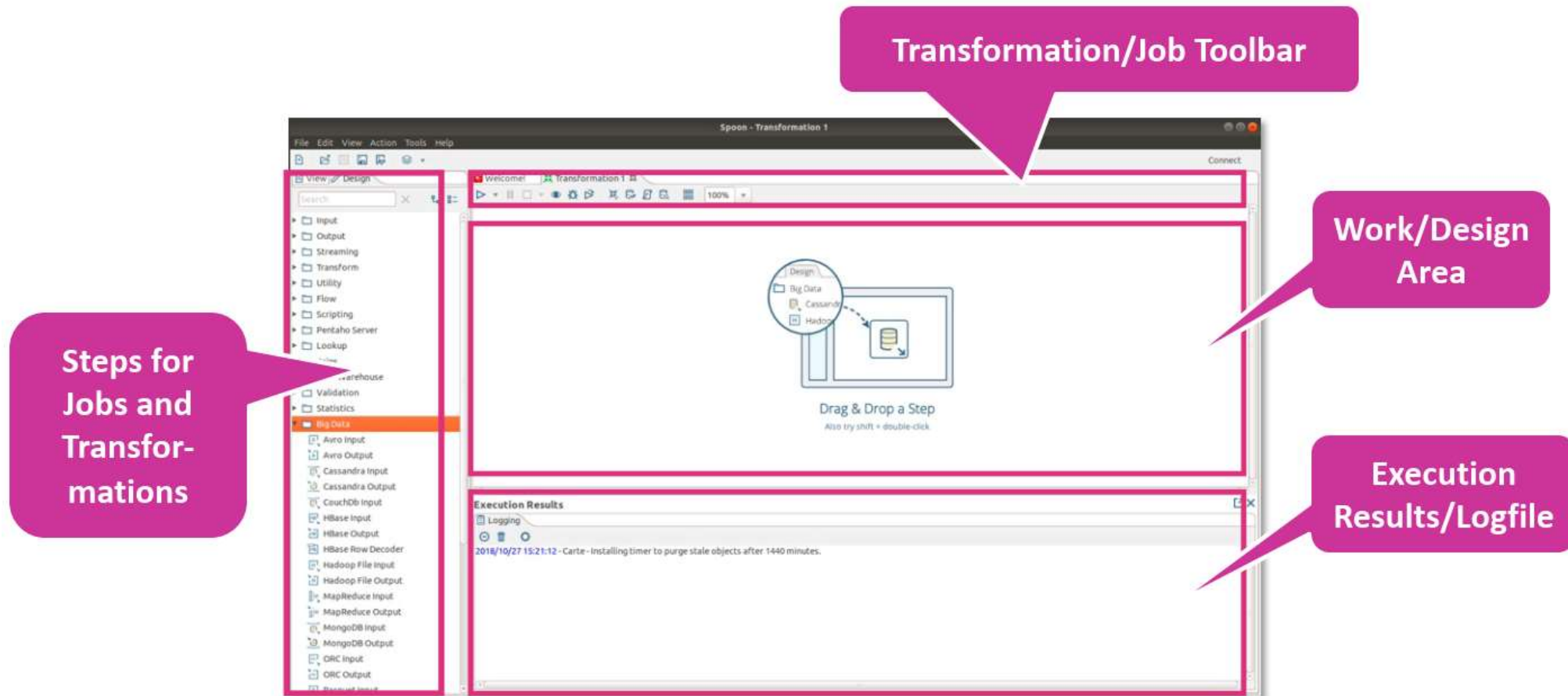
```
deb http://cz.archive.ubuntu.com/ubuntu bionic main universe
```

```
sudo apt-get update
```

```
sudo apt-get install libwebkitgtk-1.0-0
```

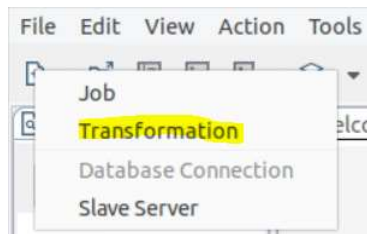

Spoon Interface

3. Запуск Pentaho Data Integration



3. Start Pentaho Data Integration

1. Создать новый **Transformation** (File >New >Transformation)

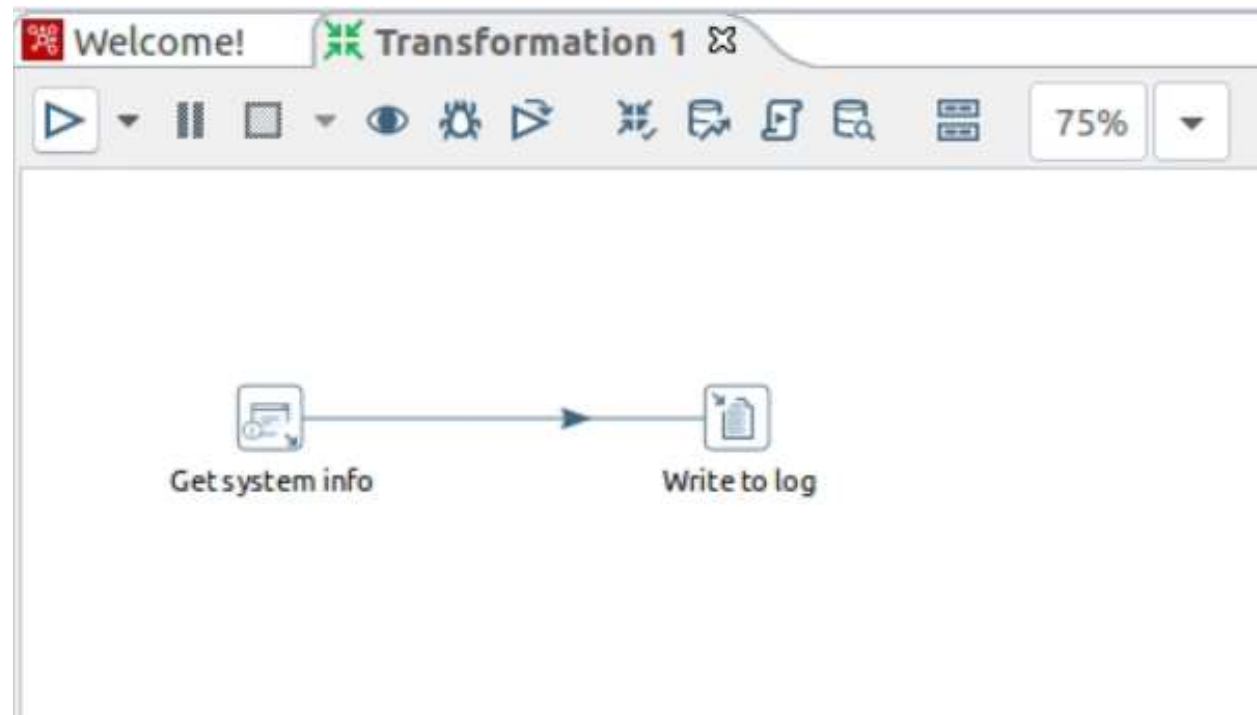
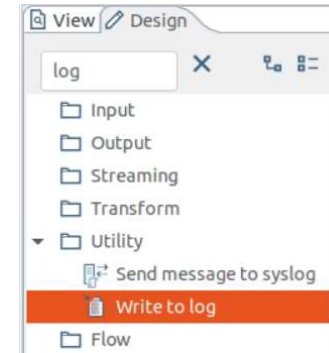
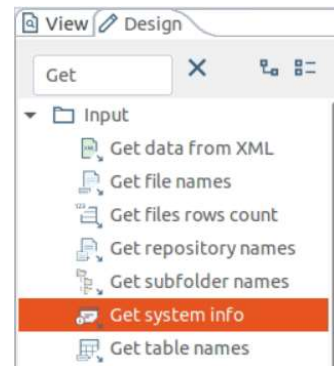


2. Перетащить (в поисковике набрать LOG) „**Get System Info**” и „**Write To Log**” в рабочую область **Work/Design Area** и соединить две задачи.

3. Соединить две задачи:



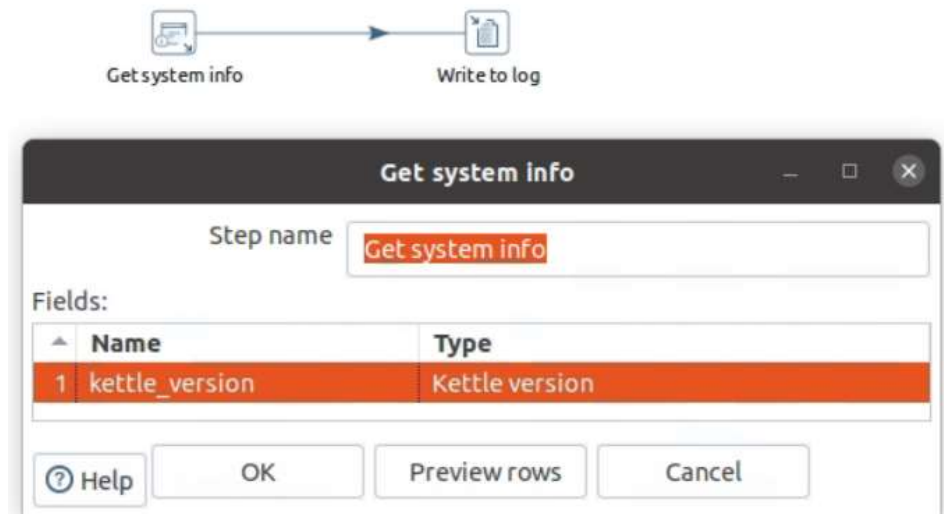
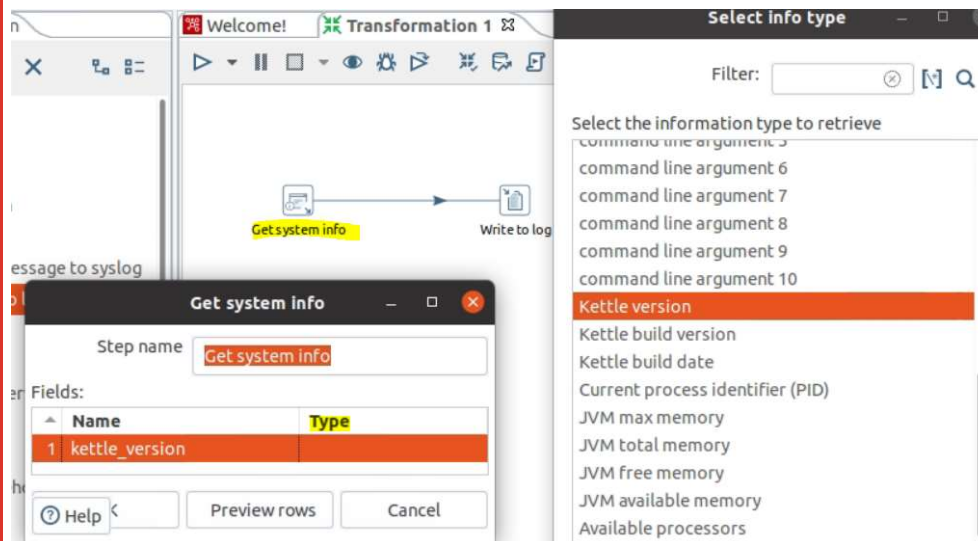
Spoon Interface



Spoon Interface

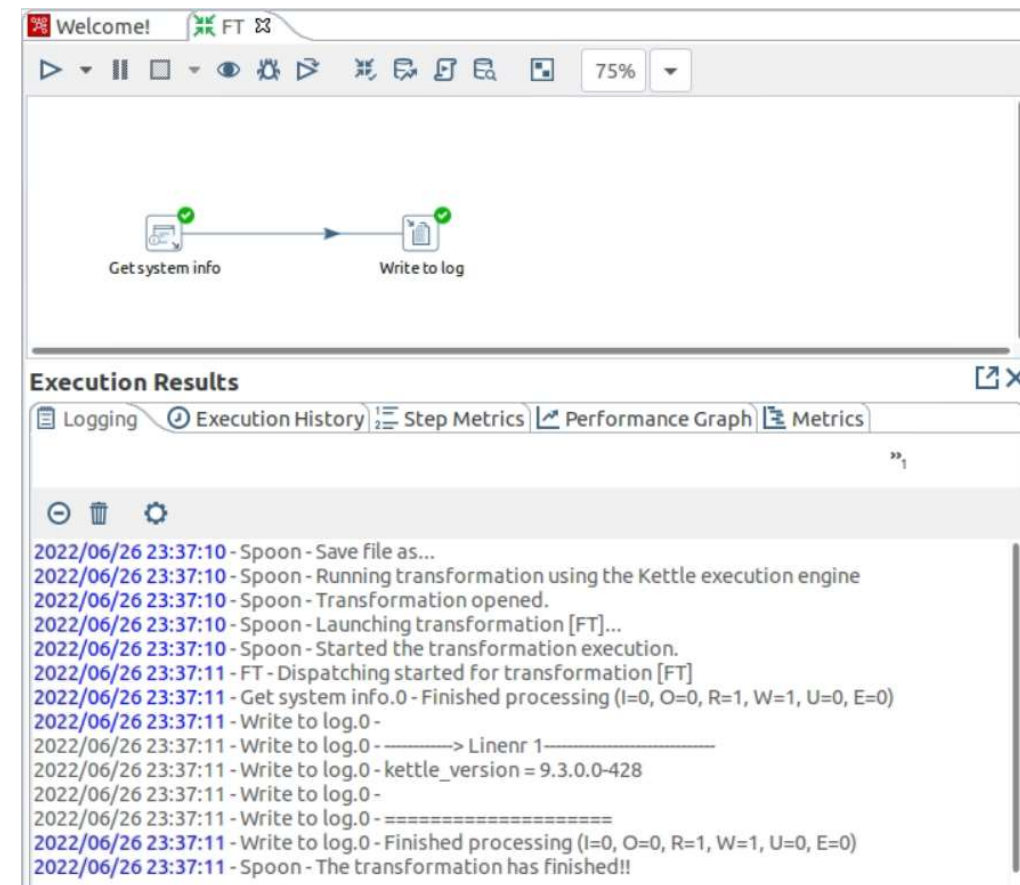
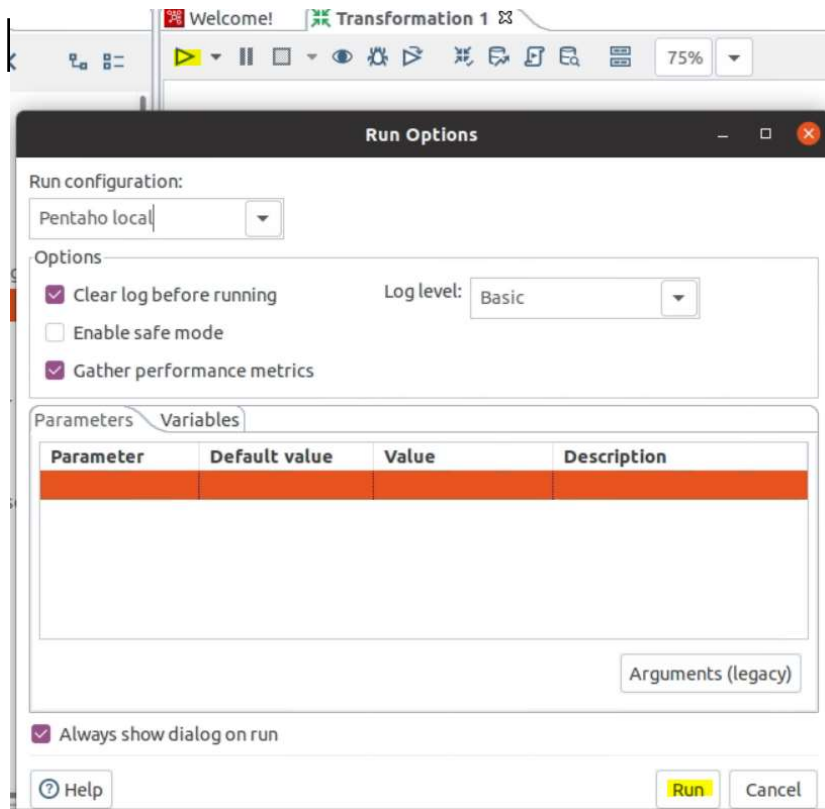
3. Start Pentaho Data Integration

4. Дважды щелкните „Get System Info“, чтобы настроить шаг в соответствии с информацией о версии (PDI):

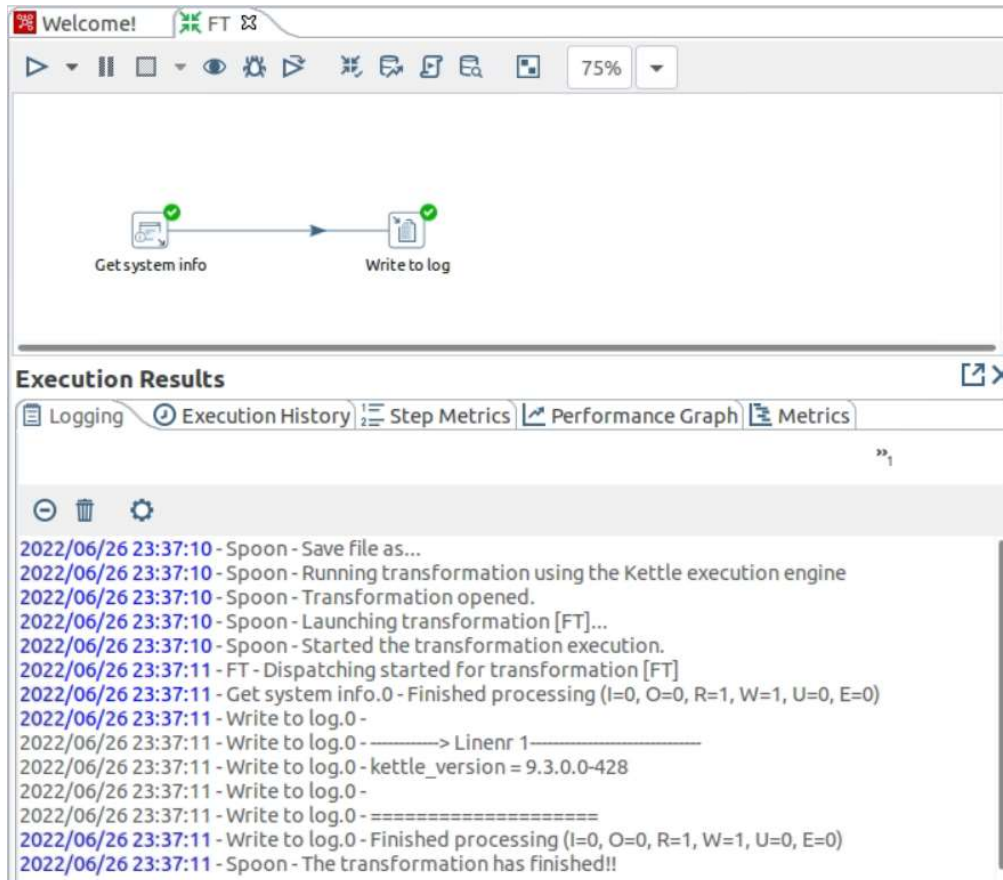


PDI Basics/Examples – First Transformation

4. Сохраните Transformation(First_Transformation).
Нажмите кнопку запуска.



PDI Basics/Examples – First Transformation

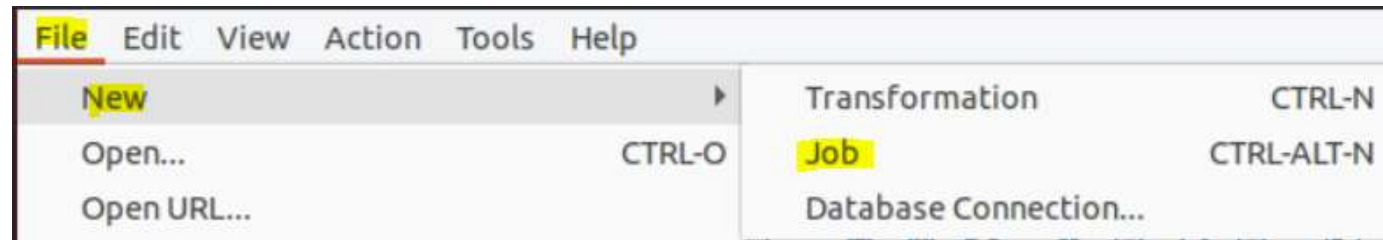


The screenshot shows the Kettle (PDI) interface. At the top, there's a 'Welcome!' window and a 'FT' tab. Below the toolbar, a transformation canvas displays two steps: 'Get system info' and 'Write to log', connected by an arrow. The 'Execution Results' panel is open at the bottom, showing a log of the execution process. The log includes messages such as 'Spoon - Save file as...', 'Spoon - Running transformation using the Kettle execution engine', 'Spoon - Transformation opened.', 'Spoon - Launching transformation [FT]...', 'Spoon - Started the transformation execution.', 'FT - Dispatching started for transformation [FT]', 'Get system info.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)', 'Write to log.0 -', 'Write to log.0 -> Linenr 1', 'Write to log.0 - kettle_version = 9.3.0.0-428', 'Write to log.0 -', 'Write to log.0 - =====', 'Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)', and 'Spoon - The transformation has finished!!'.

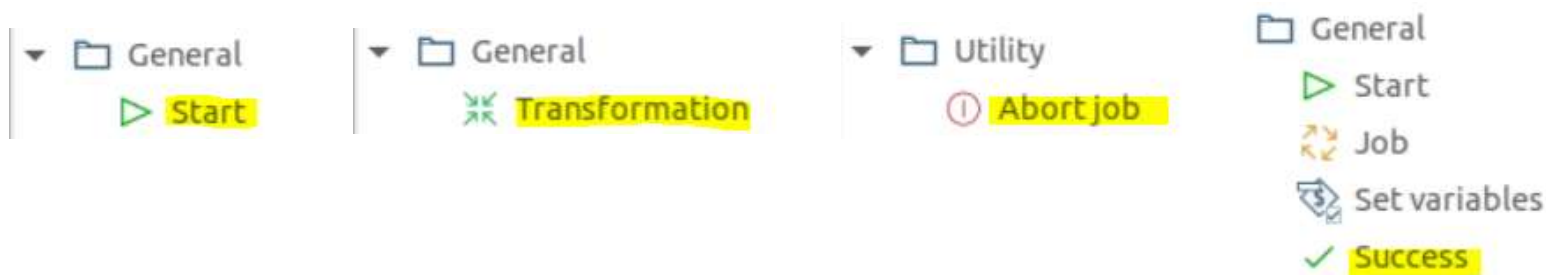
- результаты шага «Get System Info» перенаправляются на шаг «Write To Log».
- Шаг «Запись в журнал» запишет результаты в журнал выполнения.
- **transformations** связаны с фактическим преобразованием данных и потоком данных.

PDI Basics/Examples – First Job

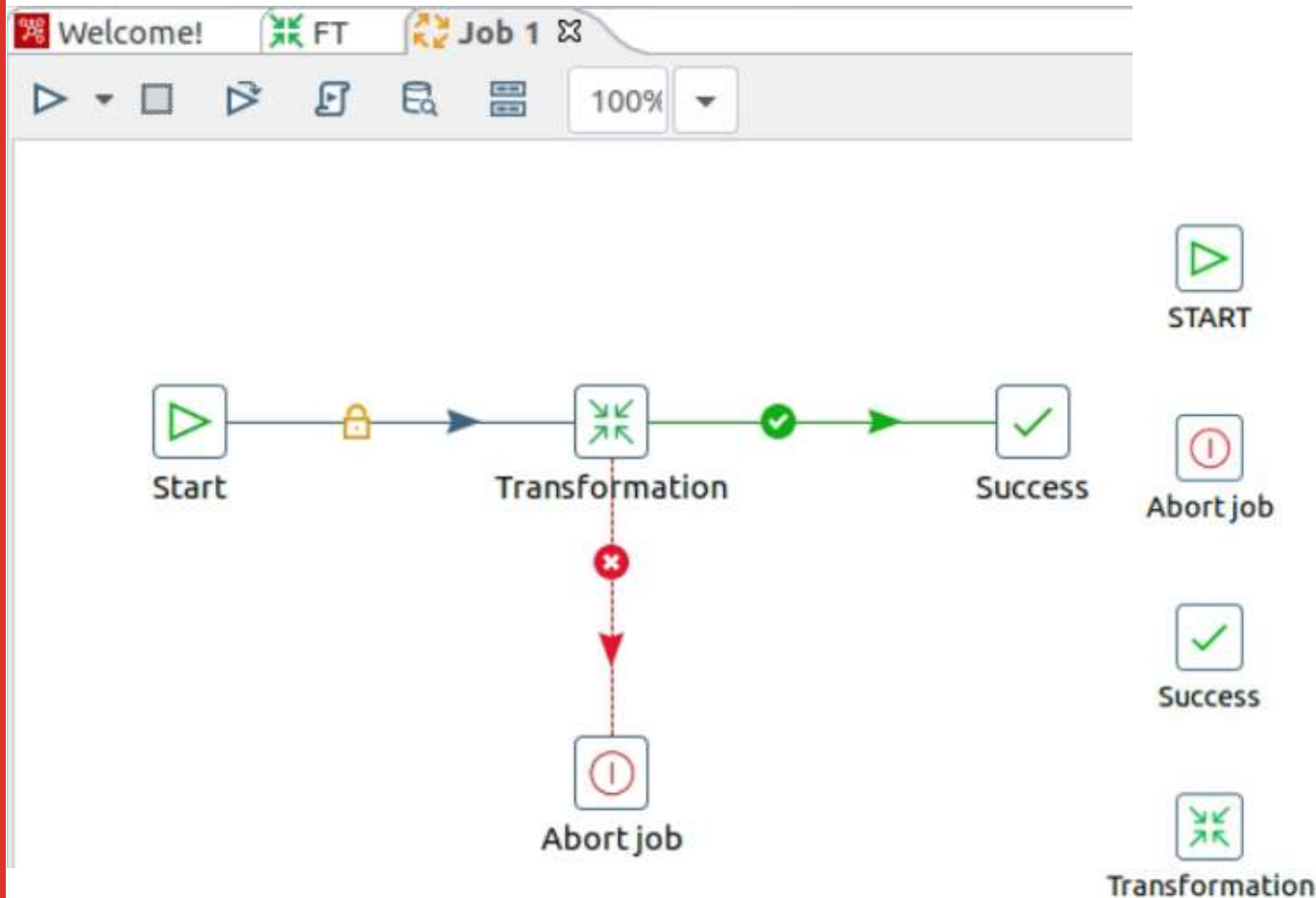
1. Создаем рабочий процесс - задание **Job** (Click: File >New >Job)



- Перетянуть „**START**“, „**Transformation**“, „**Abort Job**“ и „**Success**“ на область **Work/Design Area**, далее все шаги объединить.



PDI Basics/Examples – First Job



- Start of each Job

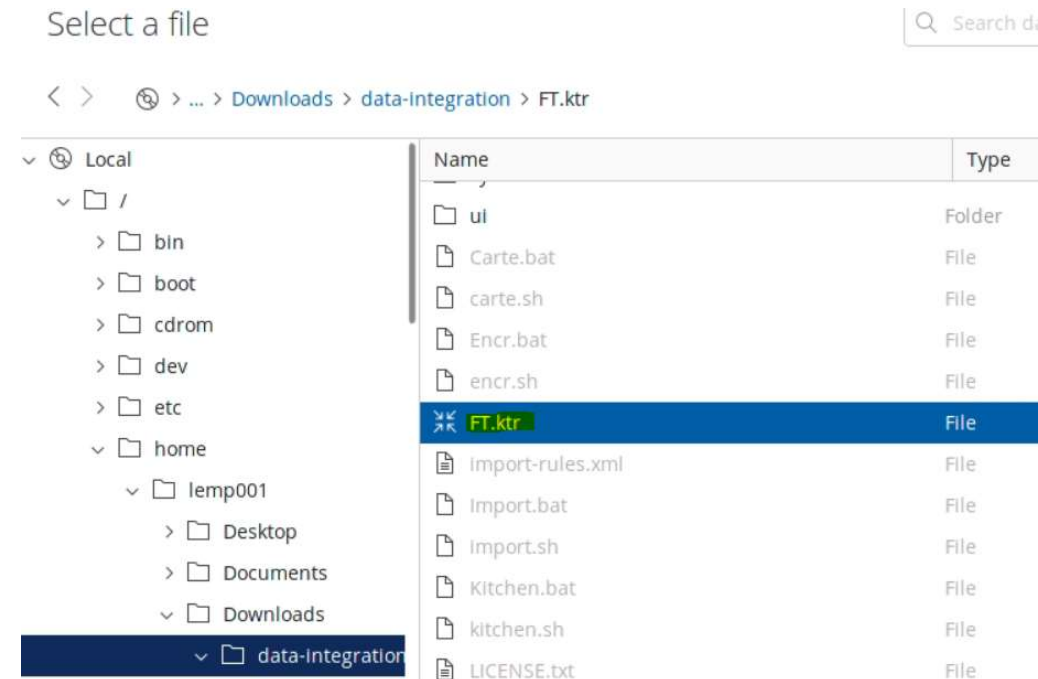
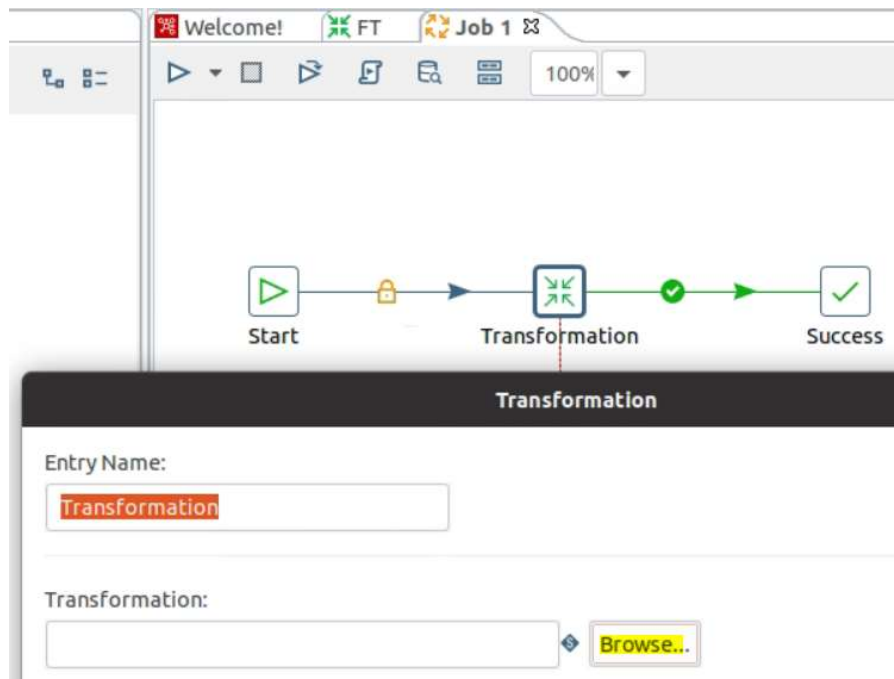
- End of a Job (if not successful)

- End of a Job (if successful)

- Includes a Transformation

PDI Basics/Examples – First Job

3. Подключить ранее созданное Преобразование (FT), дважды щелкнув по шагу „Transformation“, и включив «FT.ktr»:



PDI Basics/Examples – First Job

4. Сохранить задание (**First_Job**). Нажмите кнопку запуска.
Просмотреть результаты:

Run Options

Run configuration:
Pentaho local

Options

☐ Expand remote job Log level: Basic

☒ Clear log before running Start job at:

☐ Enable safe mode

☒ Gather performance metrics

Parameters Variables

Parameter	Default value	Value	Description
-----------	---------------	-------	-------------

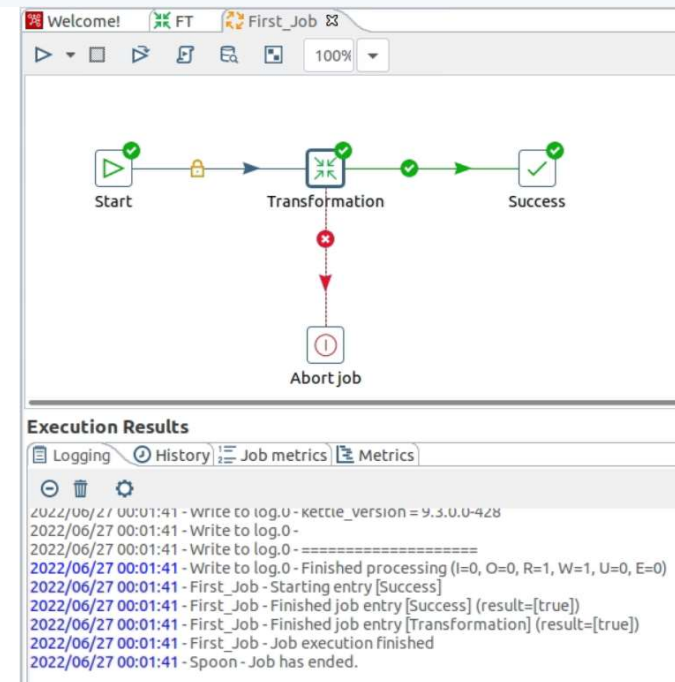
Arguments (legacy)

☒ Always show dialog on run

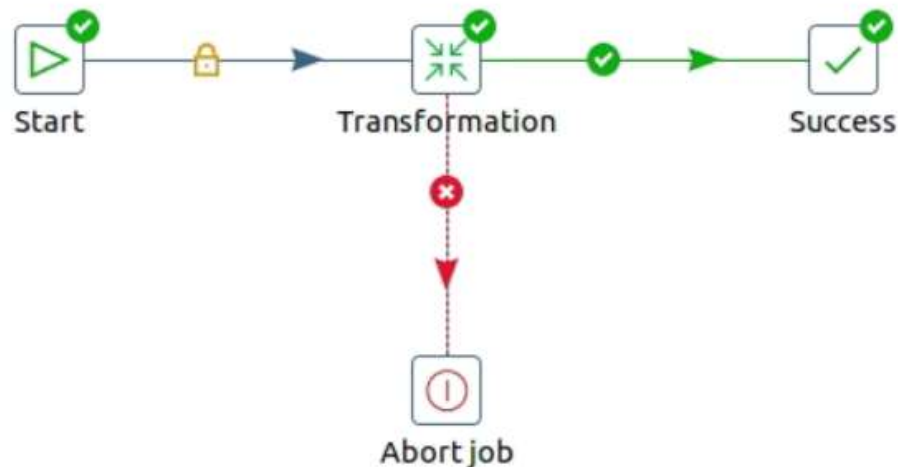
Help Run Cancel

Name: First_Job.kjb

Save in folder: home lemp001 Downloads data-integration



PDI Basics/Examples – First Job



Execution Results

Logging History Job metrics Metrics

2022/06/27 00:01:41 - Write to log.0 -
2022/06/27 00:01:41 - Write to log.0 -> Linenr 1-----
2022/06/27 00:01:41 - Write to log.0 - **kettle_version = 9.3.0.0-428**
2022/06/27 00:01:41 - Write to log.0 -
2022/06/27 00:01:41 - Write to log.0 -
2022/06/27 00:01:41 - Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/06/27 00:01:41 - First_Job - Starting entry [Success]
2022/06/27 00:01:41 - First_Job - Finished job entry [Success] (result=[true])
2022/06/27 00:01:41 - First_Job - Finished job entry [Transformation] (result=[true])
2022/06/27 00:01:41 - First Job - Job execution finished

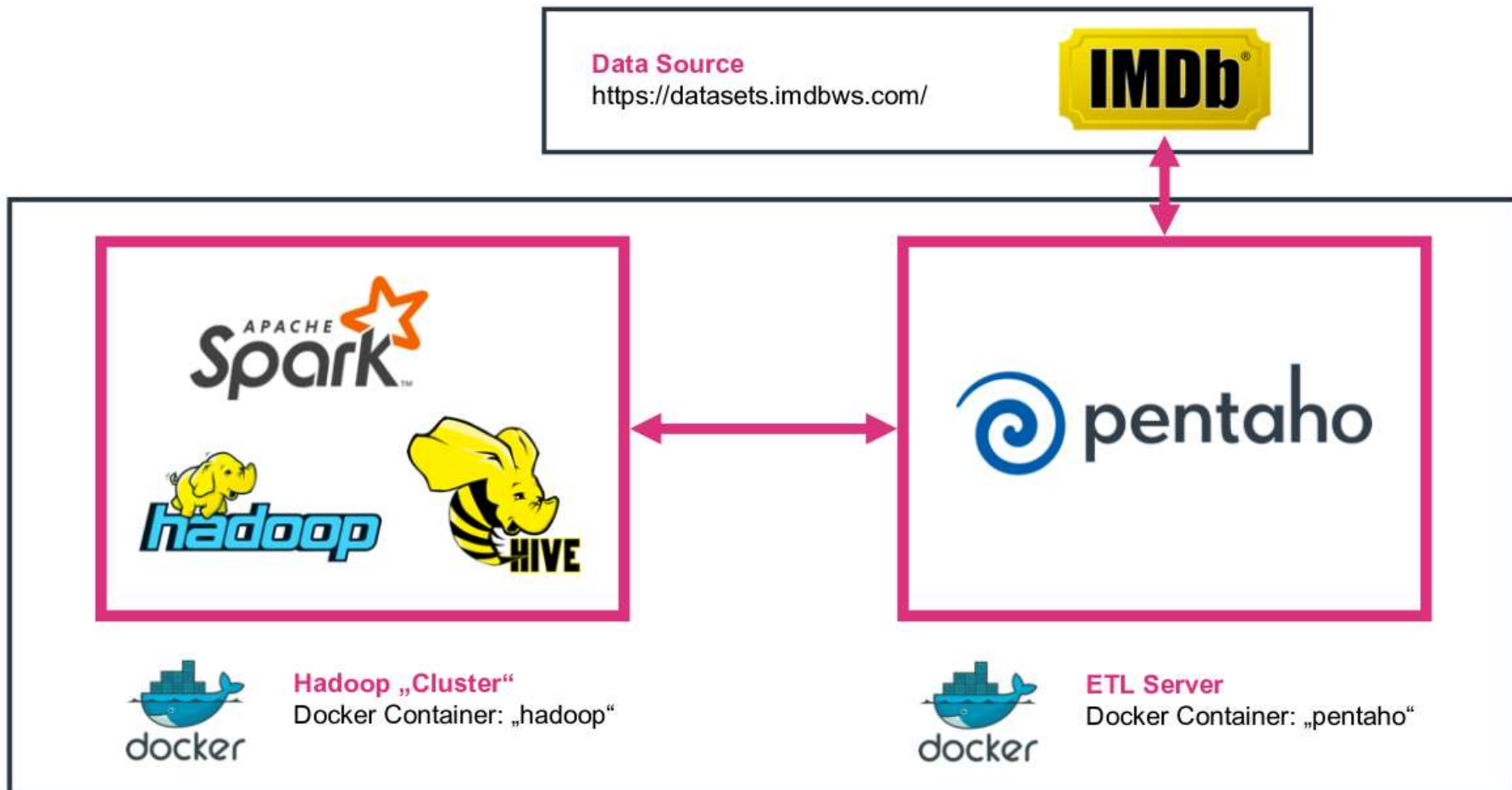
Задание(**Job**) выполнит ранее созданное преобразование(**FT.ktr**).

Результат преобразования будет добавлен к выполнению «журнал работы».

jobs связаны с рабочими процессами(**workflows**) нескольких преобразований и заданий.

Что мы хотим сделать?

The Hadoop Ecosystem



Start Hadoop/Hive/Spark Docker Container

1. Pull Docker Image:

```
sudo docker pull marcelmittelstaedt/spark_base:latest
```

2. Start Docker Image:

```
sudo docker run -dit --name hadoop \  
-p 8088:8088 -p 9870:9870 -p 9864:9864 -p 10000:10000 \  
-p 8032:8032 -p 8030:8030 -p 8031:8031 -p 9000:9000 \  
-p 8888:8888 --net bigdatanet \  
marcelmittelstaedt/spark_base:latest
```

3. Get into Docker container:

```
sudo docker exec -it hadoop bash
```

4. Switch to hadoop user:

```
sudo su hadoop  
cd
```

5. Start Hadoop Cluster:

```
start-all.sh
```

6. Start HiveServer2:

```
hiveserver2
```

Start ETL (Pentaho) Docker Container

1. Pull Docker Image:

```
sudo docker pull marcelmittelstaedt/pentaho:latest
```

2. Start Docker Image:

```
sudo docker run -dit --name pentaho \  
--net bigdatanet \  
marcelmittelstaedt/pentaho:latest
```

Wait till first Container Initialization finished:

```
sudo docker logs pentaho
```

3. Get into Docker container:

```
sudo docker exec -it pentaho bash
```

4. Switch to hadoop user:

```
sudo su pentaho  
cd
```

Execute Transformations using pan.sh

1. Run first Transformation on ETL Server:

`/home/pentaho/pentaho/data-integration/pan.sh -file=/home/pentaho/pdi_jobs/First_Transformation.ktr`

```
2022/06/26 21:59:31 - Pan - Start of run.
2022/06/26 21:59:31 - First_Transformation - Dispatching started for transformation [First_Transformation]
2022/06/26 21:59:31 - Get System Info.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/06/26 21:59:31 - Write to log.0 - 
2022/06/26 21:59:31 - Write to log.0 - -----> Linenr 1-----
2022/06/26 21:59:31 - Write to log.0 - kettle_version = 8.0.0.0-28
2022/06/26 21:59:31 - Write to log.0 - 
2022/06/26 21:59:31 - Write to log.0 - =====
2022/06/26 21:59:31 - Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/06/26 21:59:31 - Pan - Finished!
2022/06/26 21:59:31 - Pan - Start=2022/06/26 21:59:31.729, Stop=2022/06/26 21:59:31.966
2022/06/26 21:59:31 - Pan - Processing ended after 0 seconds.
2022/06/26 21:59:31 - First_Transformation - 
2022/06/26 21:59:31 - First_Transformation - Step Get System Info.0 ended successfully, processed 1 lines. ( - lines/s)
2022/06/26 21:59:31 - First_Transformation - Step Write to log.0 ended successfully, processed 1 lines. ( - lines/s)
```


Execute Transformations using pan.sh

1. Run first Job on ETL Server:

`/home/pentaho/pentaho/data-integration/kitchen.sh -file=/home/pentaho/pdi_jobs/First_Job.kjb`

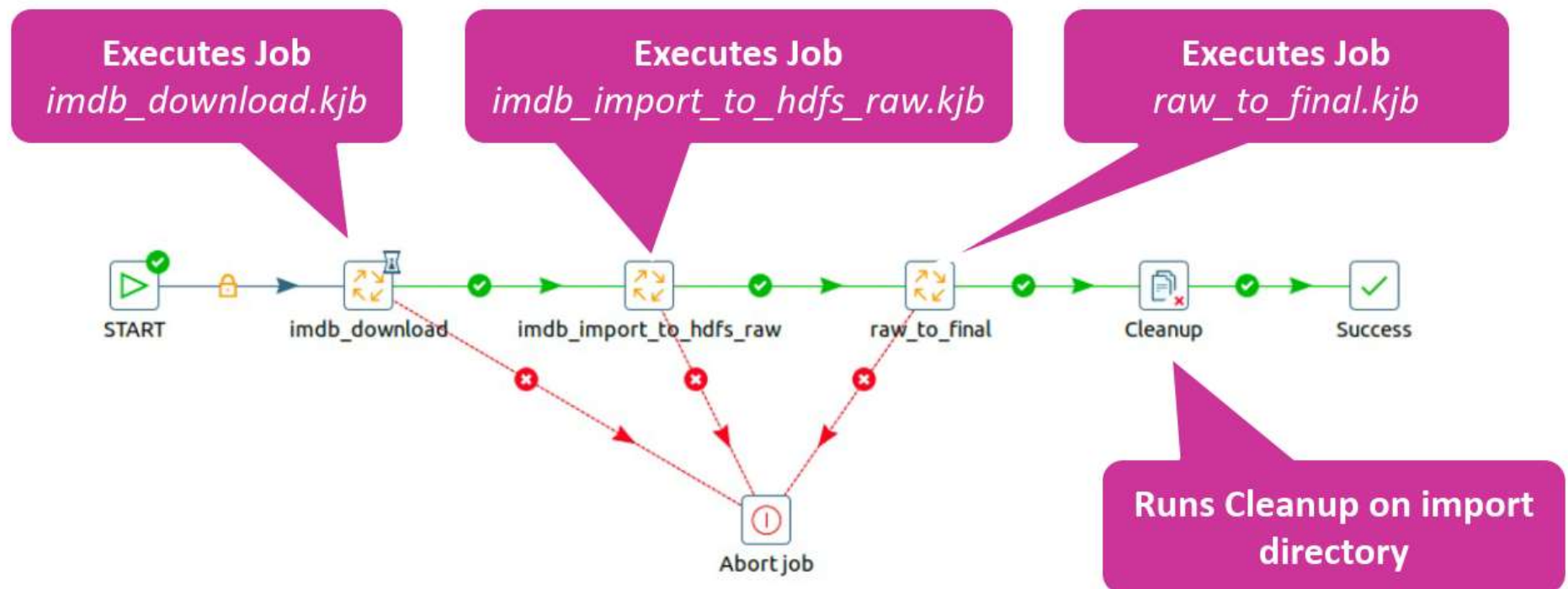
```
2022/06/26 22:09:48 - Transformation - Using run configuration [Pentaho local]
2022/06/26 22:09:48 - Transformation - Using legacy execution engine
2022/06/26 22:09:48 - First_Transformation - Dispatching started for transformation [First_Transformation]
2022/06/26 22:09:48 - Write to log.0 -
2022/06/26 22:09:48 - Write to log.0 - -----> Linenr 1-----
2022/06/26 22:09:48 - Write to log.0 - kete_version = 8.0.0.0-28
2022/06/26 22:09:48 - Write to log.0 -
2022/06/26 22:09:48 - Write to log.0 - =====
2022/06/26 22:09:48 - Get System Info.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/06/26 22:09:48 - Write to log.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2022/06/26 22:09:48 - First_Job - Starting entry [Success]
2022/06/26 22:09:48 - First_Job - Finished job entry [Success] (result=[true])
2022/06/26 22:09:48 - First_Job - Finished job entry [Transformation] (result=[true])
2022/06/26 22:09:48 - First_Job - Job execution finished
2022/06/26 22:09:48 - Kitchen - Finished!
2022/06/26 22:09:48 - Kitchen - Start=2022/06/26 22:09:37.031, Stop=2022/06/26 22:09:48.762
2022/06/26 22:09:48 - Kitchen - Processing ended after 11 seconds.
```

САМОСТОЯТЕЛЬНАЯ РАБОТА

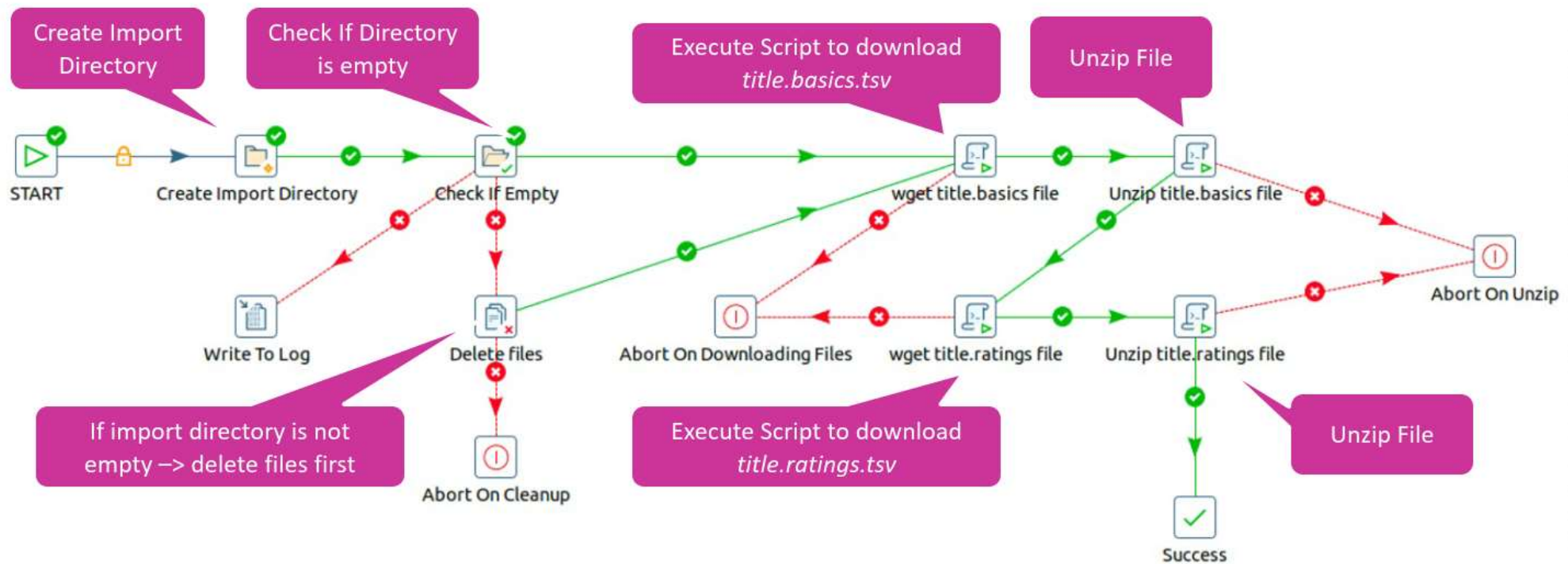
Простые примеры ETL

инструмент рабочего процесса ETL (PDI) для интеграции данных IMDb

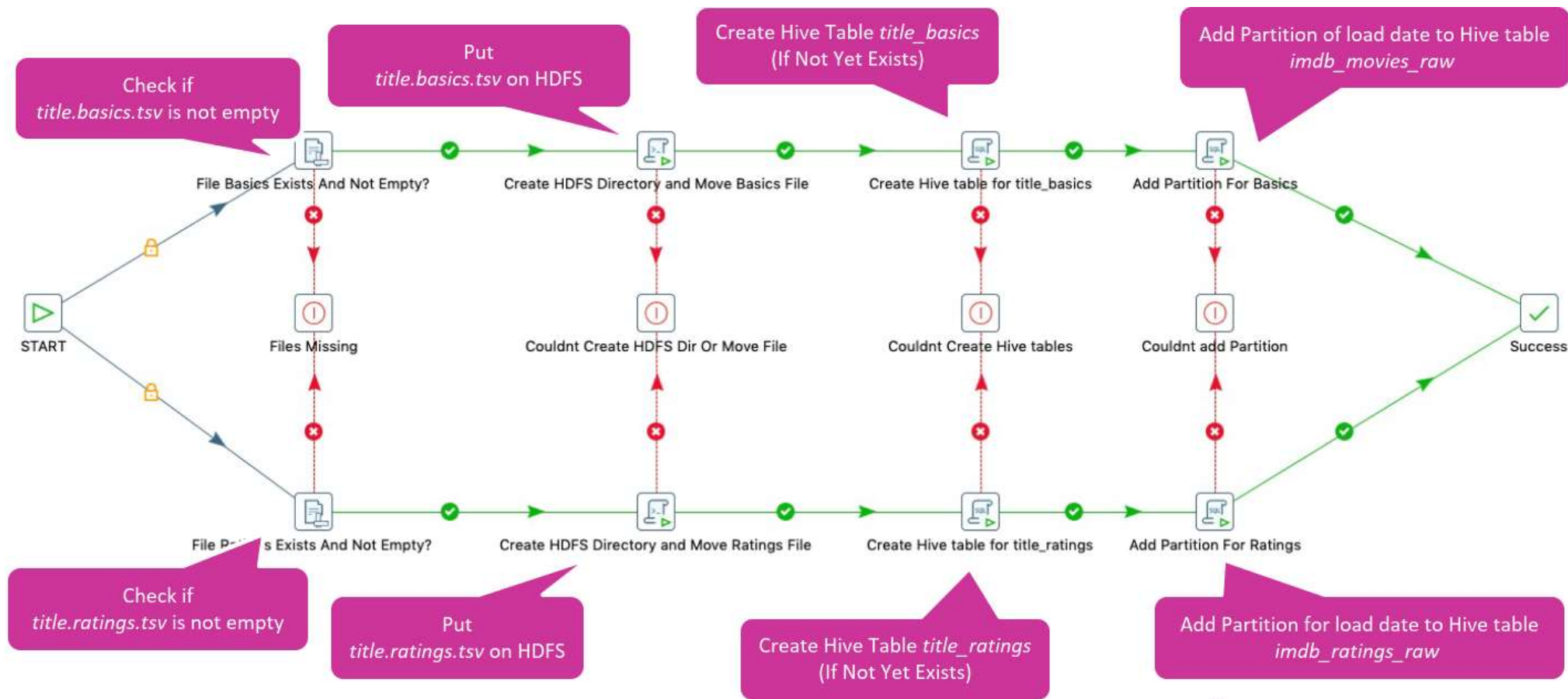
PDI IMDb Import – Main Job (imdb_main.kjb)



PDI IMDb Import – imdb_download.kjb



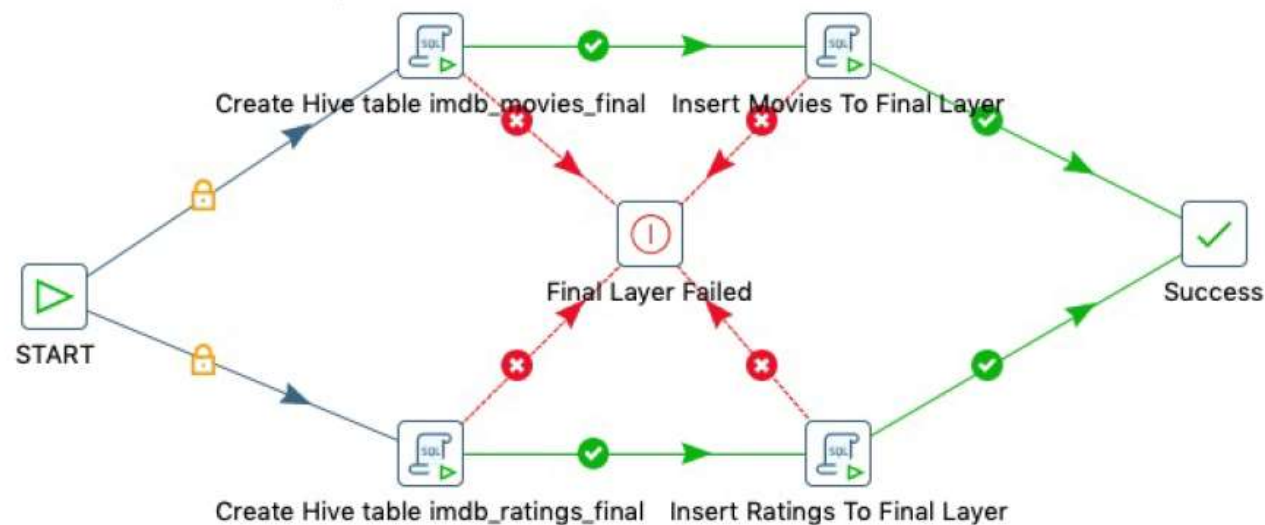
PDI IMDB Import – imdb_import_to_hdfs_raw.kjb



PDI IMDB Import – raw_to_final.kjb

Create Hive Table *imdb_movies*
(If Not Yet Exists)

Fill table *imdb_movies* by using:
- raw layer table *title_basics* and
- **Dynamic** partitioning



Create Hive Table *imdb_ratings*
(If Not Yet Exists)

Fill table *imdb_ratings* by using:
- raw layer table *title_ratings* and
- **Static** partitioning

СПАСИБО ЗА ВНИМАНИЕ