

# Pentaho Data Integration

Enterprise Solution

**James O'Reilly**

Hitachi Vantara Global Learning  
Date



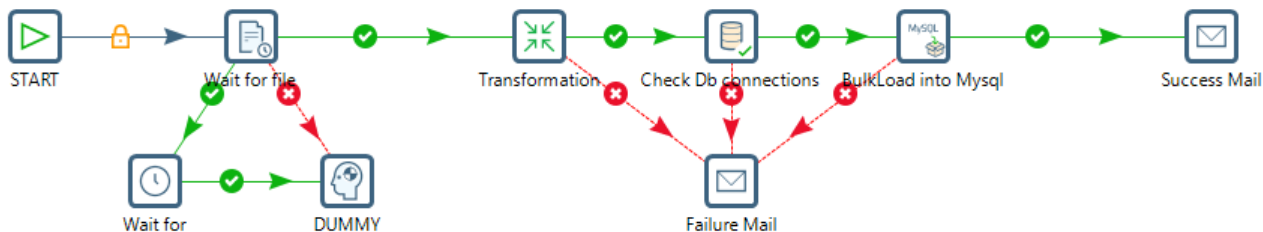
# Module Objectives

When you complete this module, you should be able to:

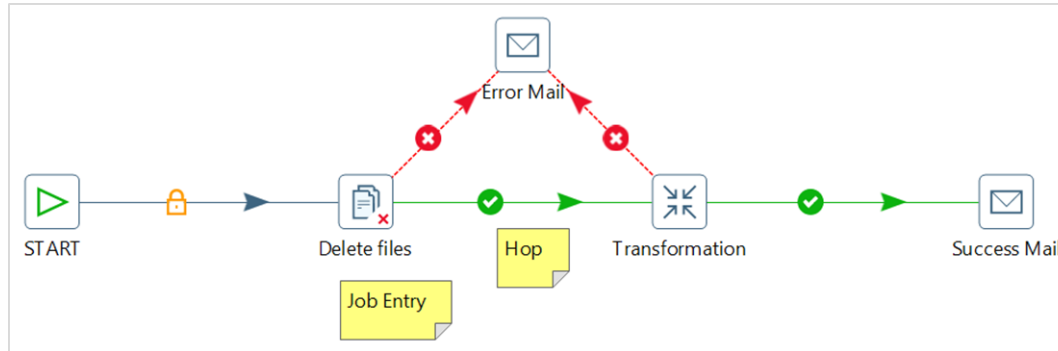
- Implement Jobs to control your ETL workflow
- Deploy your solution with Variables and Parameters
- Scale out your solution with Carte Servers

# Jobs

# Jobs Workflow



- Adding and configuring Job Entries
- Ensure required resources are available
- Using file management Job Entries
- Implement error handling in Jobs
- Implement Checkpoints
- Reviewing execution results



- Jobs are workflow-like models for coordinating resources, execution and dependencies of ETL activities
- Consist of job entries, hops and notes
- Aggregate individual transformations to a process and introduce order
- Hops can be conditional

- In an IT environment, orchestration is the ongoing task of ensuring that technical processes run reliably and according to schedule while producing the correct outputs using acceptable amounts of resources
- Orchestration includes:
  - Ensuring the availability of resources
  - Executing a series of tasks in the right order
  - Detecting and recovering from errors
  - Logging the results
  - Notifying people and other applications

# Job Management

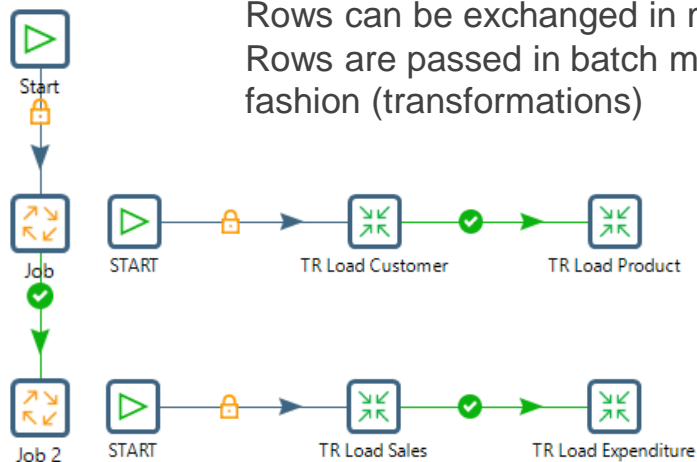
- Always check the status of external systems
  - Host systems
  - Databases / tables
  - File systems / directories
  - FTP sites
  - Web services
- Execute tasks with retry logic
- Notify others when a problem has occurred
- Store variables in a central location and use them consistently

# Job Entry

- A job (Job / Transformation) entry is the primary building block of a job.
  - Execute jobs / transformations, retrieve files, generate email, and so on
- Each job can only have one job entry.

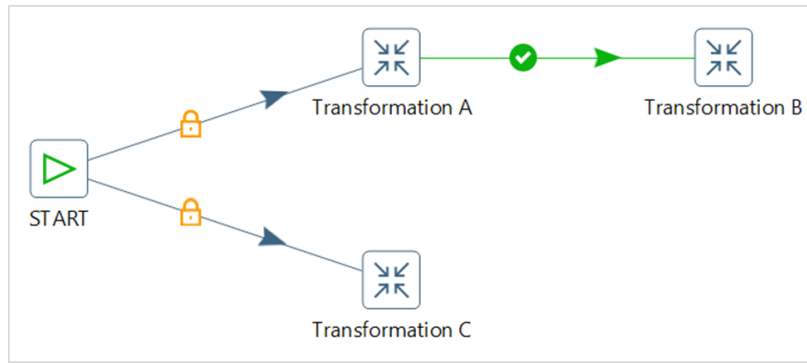
## Job entries can pass a result object containing rows

Rows can be exchanged in memory – result rows  
Rows are passed in batch mode, not in a streaming fashion (transformations)





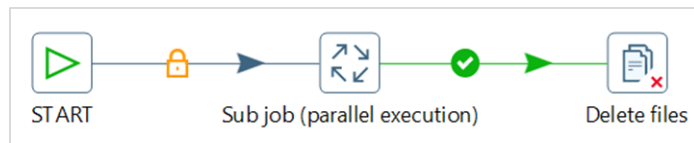
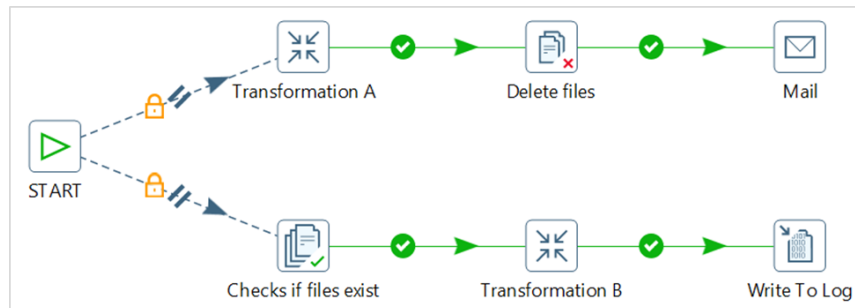
# Multiple Paths & Backtracking



- Jobs are executed using a backtracking algorithm
- The path itself is defined by the actual outcome (success or failure) of the job entries
- The path is always followed until the very end before a next possibility is considered
- Order depends on creation of job entries
- The result of the job (success or failure) depends on the last job entry

# Parallel Execution

- A job entry can be told to execute the next job entries in parallel
  - Running in separate threads
- When you have a number of sequential job entries, these are executed in parallel as well

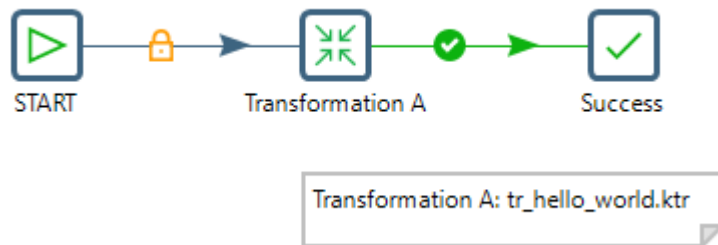


- If you want to continue in sequence after this parallel execution you need to put the parallel part of the job in a sub-job

# Lab 1 – Create a Job

# Lab 1 - Hello World - Job

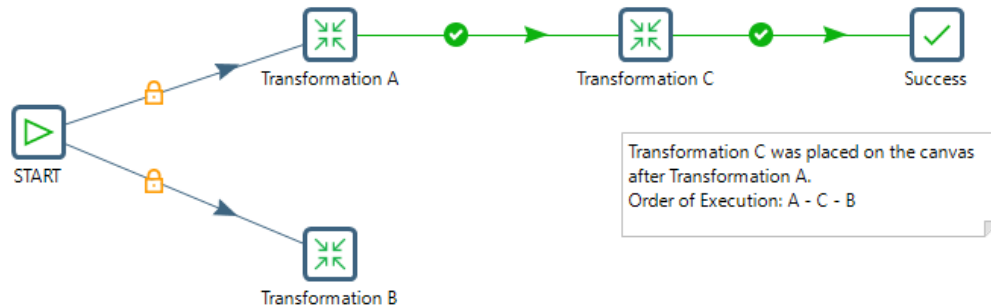
- Simple Job



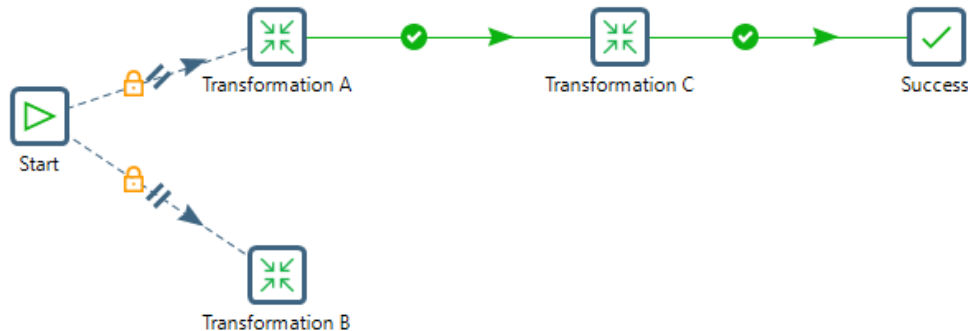
# Lab 2 – Order of Execution

# Lab 2 – Order of Execution

- Backward Algorithm



- Parallel



# Variables & Parameters

# Variables, Parameters

When you Run a Transformation or Job, there are several ways you can define the required settings.

- Variables allow you to dynamically enter the required setting.
- Parameters statically sets the condition of the Transformation or Job.



- Variables can be used throughout Pentaho Data Integration, including in transformation steps and job entries. You define variables by setting them with the Set Variable step in a transformation or by setting them in the kettle.properties file in the directory:
  - \$HOME/.kettle (Unix/Linux/OSX)
  - C:\Documents and Settings\<username>\.kettle\ (Windows XP)
  - C:\Users\<username>\.kettle\ (Windows Vista, 7 and later)
- The way to use them is either by grabbing them using the Get Variable step or by specifying meta-data strings like:
  - `${VARIABLE}`
  - or:
  - `%%VARIABLE%%`

- Parameters are special in the sense that they are explicitly named command line arguments. If you pass on a lot of arguments to your Kettle Job or Transformation, it might help to assign those values to an explicitly named parameter.
- Parameters have following advantages:
  - On the commandline you assign the value directly to a parameter, hence there is zero chance of a mix-up.
  - A default value can be defined for a named parameter
  - A description can be provided for a named parameter
  - No need for an additional transformation that sets the variables for the job

# Lab 1 – Parameters & Variables

# Lab 1 - Variables and Parameters

- So lets modify an existing Transformation ..
- Open Module 3 / Lab 2: Reading from a Database Table

The 'Table input' dialog box is shown. It has a 'Step name' field set to 'Table input' and a 'Connection' dropdown set to 'Sampledata'. There are buttons for 'Edit...', 'New...', and 'Wizard...'. Below these is a 'SQL' section with a 'Get SQL select statement...' button. The SQL text area contains the following query:

```
SELECT  
  ORDERNUMBER  
  , ORDERDATE  
  , REQUIREDDATE  
  , SHIPPEDDATE  
  , STATUS  
  , COMMENTS  
  , CUSTOMERNUMBER  
FROM ORDERS  
WHERE STATUS = '${STATUS}'
```

Below the SQL text area, there is a section for 'Line 1 Column 0' with the following options:

- ☐ Enable lazy conversion
- ☒ Replace variables in
- Insert data from step:
- ☐ Execute for each row?
- Limit size:

At the bottom are buttons for '? Help', 'OK', 'Preview', and 'Cancel'.

The 'Transformation properties' dialog box is shown, with the 'Parameters' tab selected. It contains a table with the following data:

#	Parameter	Default Value	Description
1	STATUS	Shipped	

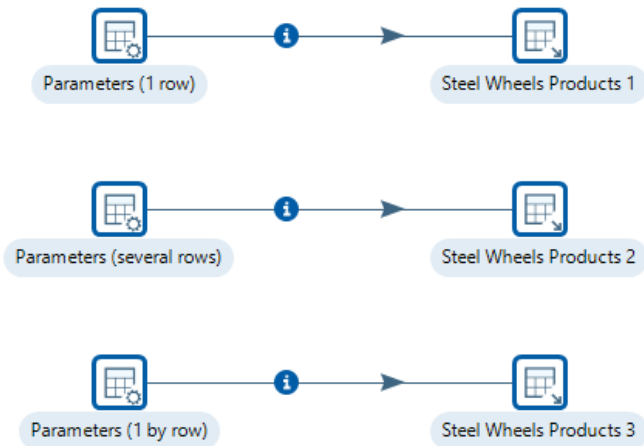
At the bottom are buttons for 'OK', 'SQL', and 'Cancel'.

# Lab 2 – Parameters and SQL

# Lab 2 – Parameters in SQL

- One of the ways you have to make your queries more flexible is by passing it through some parameters.

This demonstration illustrates the different options for passing parameters.



The image shows two screenshots of the 'Add constant rows' dialog box, illustrating different parameter configurations.

**Top Screenshot:** The 'Step name' is 'Parameters (1 by row)'. The 'Meta' tab is selected, showing a table with one row: # 1, Name parameter, Type String.

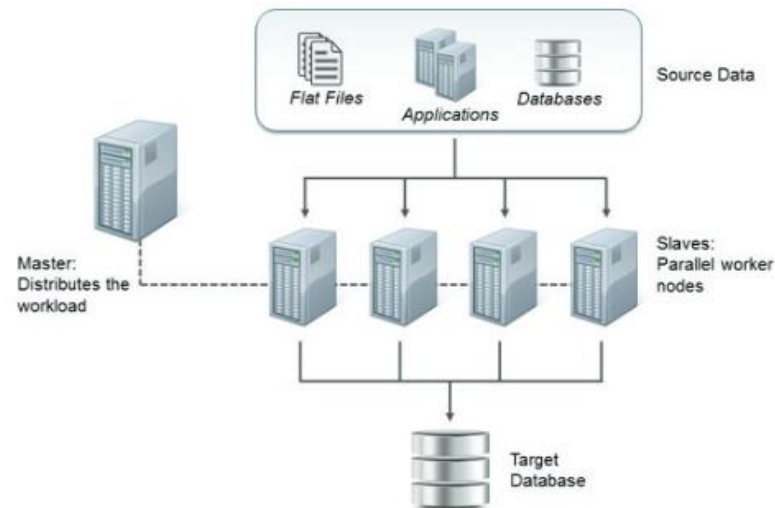
**Bottom Screenshot:** The 'Step name' is 'Parameters (1 by row)'. The 'Meta' tab is selected, showing a table with two rows: # 1, Name Classic Cars, Type String; # 2, Name 1:10, Type String.

Both screenshots show the 'OK', 'Preview', and 'Cancel' buttons. The bottom screenshot also shows the 'Help' button.

# Enterprise Scaling

# Clustering Carte Slave servers

- Clustering is a technique that can be used to scale out transformations to make them run on multiple servers, in parallel.
- The Cluster is defined with a Schema, with one master server acting as the controller for the cluster.
- The Cluster schema also contains metadata on how master and slaves pass data back and forth.





# Carte Configuration

## Master DI server

- Default configuration for low volume and process mix

## Master DI server with Independent number Carte servers

- Used for large amount of transformations of lesser volume
- Master acts as a load balancer for transformations

## Master DI server with slave servers (Clustering)

- Used for large volumes with fewer transformations
- Distributes set of rows across slave servers
- Can be fixed cluster size or dynamic

# Carte as a Slave Server

- Carte is a lightweight HTTP server that accepts commands from remote clients.
- These commands control the deployment, management, and monitoring of Jobs and Transformations.
- You can configure the carte server with an XML file (examples found in \pwd directory)

# Lab 1 – Implement Carte Servers

# Lab 1 – Start Master / Slave Nodes

- Script to start a Master and Slave node
  - Topic 3 - Scalability\Lab 1 - Master - Slave Nodes\config\_nodes.cmd

```
C:\WINDOWS\system32\cmd.exe

This script starts a Pentaho Cluster.

1. Master Server - p 11000
   wait until Master node is in listening mode

2. Slave Node 1 - p 11100

Select an Option:
```

```
C:\WINDOWS\system32\cmd.exe - carte localhost 11000

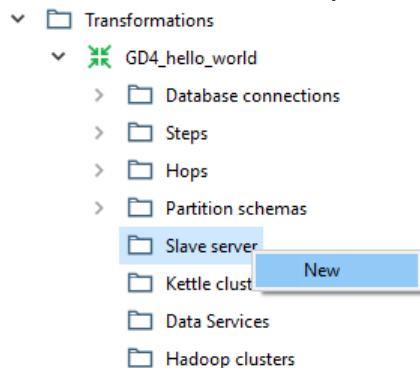
INFO: [KnowledgeFlow] Loading properties and plugins...
2019-09-03 14:30:55 weka.gui.beans.KnowledgeFlowApp init
INFO: [KnowledgeFlow] Initializing KF...
2019-09-03 14:30:55 weka.gui.GenericObjectEditor registerEditors
INFO: ---Registering Weka Editors---
2019-09-03 14:30:55 weka.experiment.DatabaseUtils initialize
WARNING: Trying to add database driver (JDBC): jdbc.idbDriver - Warning, not in CLASSPATH?
2019-09-03 14:30:55 weka.gui.beans.BeansProperties loadProperties
INFO: [KnowledgeFlow] Loading properties and plugins...
2019-09-03 14:30:55 weka.gui.beans.KnowledgeFlowApp init
INFO: [KnowledgeFlow] Initializing KF...
2019/09/03 14:30:55 - Carte - Installing timer to purge stale objects after 1440 minutes.
2019/09/03 14:30:56 - Carte - Created listener for webserver @ address : localhost:11000
```

```
C:\WINDOWS\system32\cmd.exe - carte localhost 11100

INFO: [KnowledgeFlow] Loading properties and plugins...
2019-09-03 14:38:34 weka.gui.beans.KnowledgeFlowApp init
INFO: [KnowledgeFlow] Initializing KF...
2019-09-03 14:38:34 weka.gui.GenericObjectEditor registerEditors
INFO: ---Registering Weka Editors---
2019-09-03 14:38:35 weka.experiment.DatabaseUtils initialize
WARNING: Trying to add database driver (JDBC): jdbc.idbDriver - Warning, not in CLASSPATH?
2019-09-03 14:38:35 weka.gui.beans.BeansProperties loadProperties
INFO: [KnowledgeFlow] Loading properties and plugins...
2019-09-03 14:38:35 weka.gui.beans.KnowledgeFlowApp init
INFO: [KnowledgeFlow] Initializing KF...
2019/09/03 14:38:35 - Carte - Installing timer to purge stale objects after 1440 minutes.
2019/09/03 14:38:35 - Carte - Created listener for webserver @ address : localhost:11100
```

# Lab 1 - Master DI Carte Server

- Open tr\_hello\_world transformation
- On the View tab, select: Slave > New



The 'Slave Server dialog' window is shown with the 'Service' tab selected. The fields are as follows:

- Server name: master
- Hostname or IP address: localhost
- Port (empty is port 80): 11000
- Web App Name (required):
- Username: cluster
- Password: (masked with dots)
- Is the master: ☒
- Use https protocol: ☐

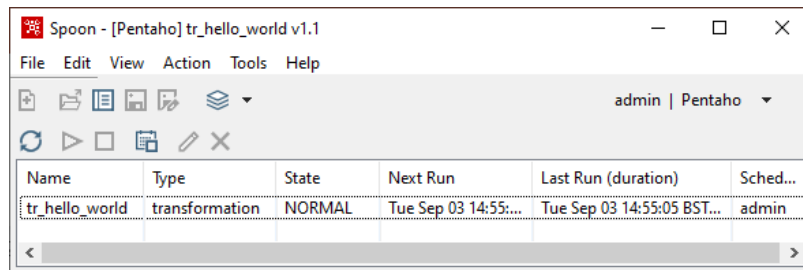
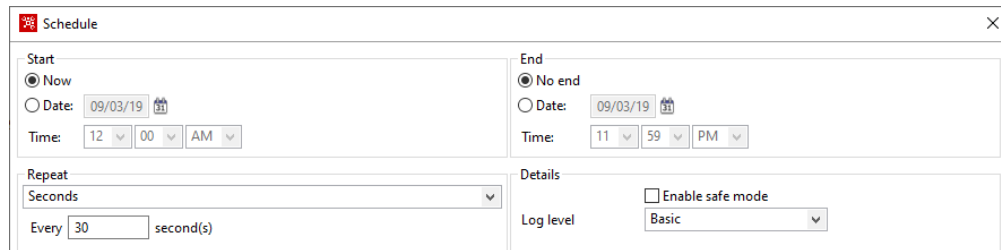
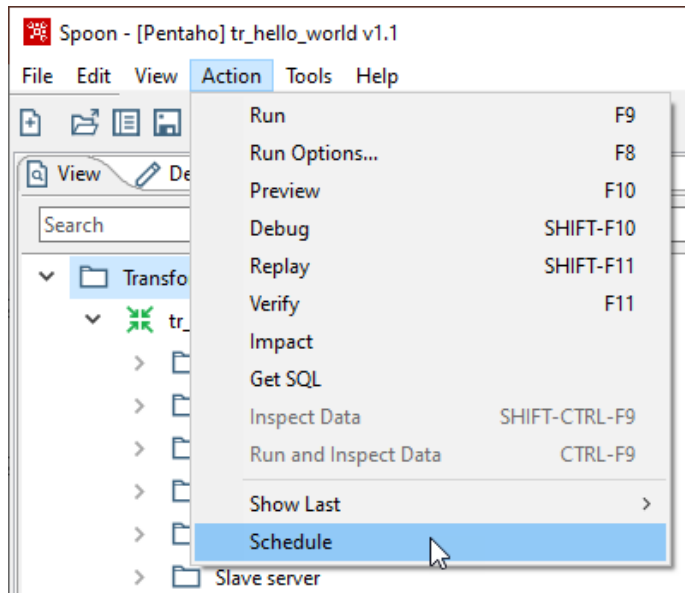
Buttons: OK, Cancel

- Right mouse click and select: Monitor
- Ensure that you have connected to the server.
- Access the server: <http://localhost:8080/kettle/status>

Username: cluster  
Password: cluster

# Lab 1 - Monitor & Schedule

## ■ Export to Repository



# Module Recap

In this module, you should have learned to:

- Implement Jobs to control your ETL workflow
- Deploy your solution with Variables and Parameters
- Scale out your solution with Carte Servers

# Thank You





**HITACHI**  
Inspire the Next 