

# Data flows in Azure Data Factory – endlich auch hier Transformationen!



Stefan Kirner

20.2.2018

## About me

### Stefan Kirner



- › PASS Chapter Lead Karlsruhe [ski@sqlpass.de](mailto:ski@sqlpass.de)
- › Teamlead BI Solutions @inovex
- › MCSE for Data Management & Analytics & Cloud Infrastructure
- › Microsoft P-TSP Data Platform
- › Twitter: @KirnerKa

inovex



inovex ist ein IT-Projekthaus  
mit dem Schwerpunkt „Digitale Transformation“:

Product Ownership · Datenprodukte  
Web · Apps · Smart Devices · BI  
Big Data · Data Science · Search  
Replatforming · Cloud · DevOps  
Data Center Automation & Hosting  
Trainings · Coachings

inovex gibt es in Karlsruhe · Pforzheim ·  
Stuttgart · München · Köln · Hamburg

Und natürlich unter [www.inovex.de](http://www.inovex.de)

Wir nutzen Technologien,  
um unsere Kunden glücklich zu machen.  
*Und uns selbst.*

# Agenda

- Intro Data Factory v2
- Control Flow & Triggers
- Data Flow
- Q+A



Intro

Data Factory v2

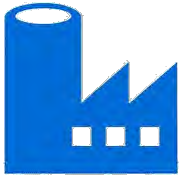
## New Pipeline Model

Rich pipeline orchestration

Triggers – on-demand, schedule, event

## Data Factory

*Managed Data Integration Service*



## Data Movement as a Service

Cloud, Hybrid

30 connectors provided

Data flow as NEW Data Transformation Layer

## SSIS Package Execution

In a managed cloud environment

Use familiar tools, SSMS & SSDT

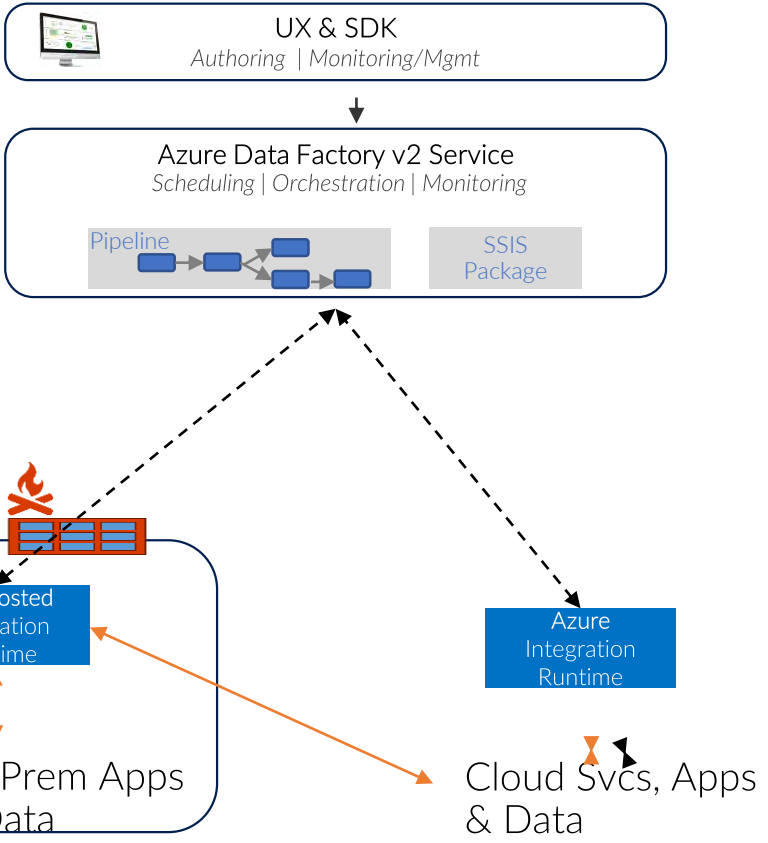
## Author & Monitor

Programmability (Python, .NET, Powershell, etc)

Visual Tools for Control Flow and **NEW: Data Flow**

←--→ Command and Control

↔ Data



## Data Factory

A data integration account.  
Location of orchestration, service metadata

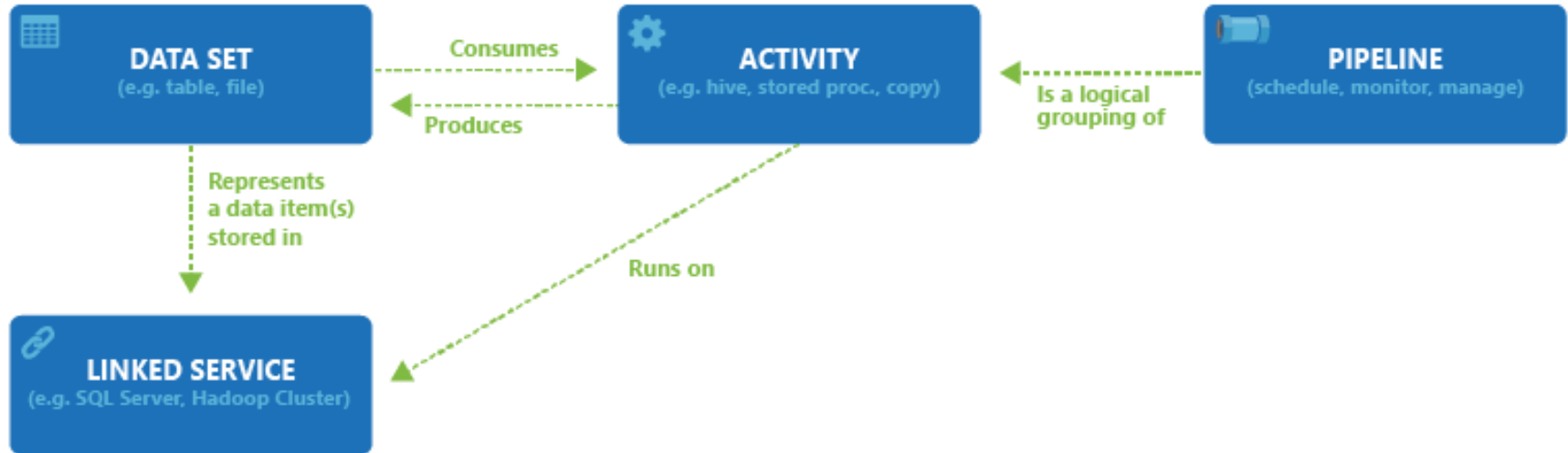
## Integration Runtime (IR)

ADF's execution engine

Three core capabilities:

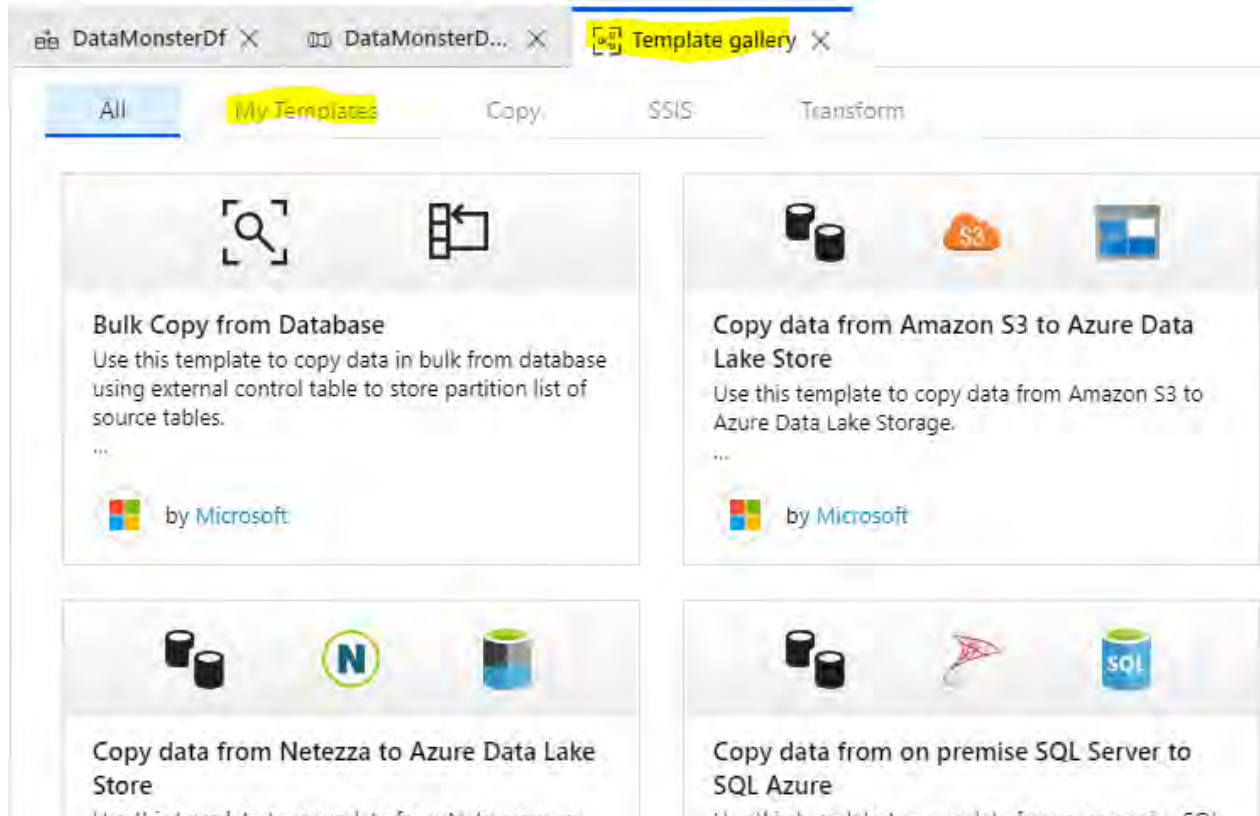
- data movement
- pipeline activity execution
- SSIS package execution

# Data Factory Essentials



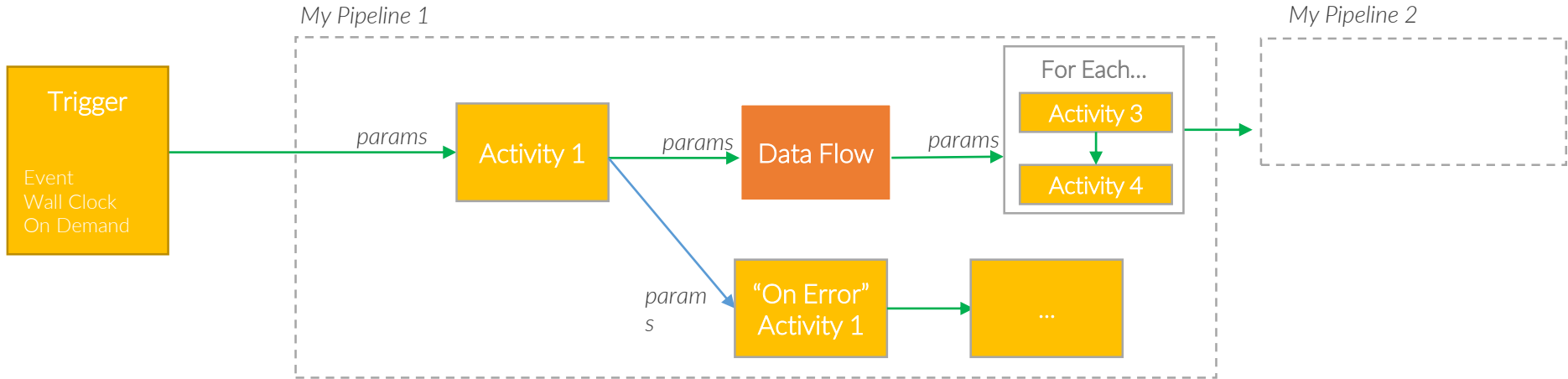


# New: Kickstart using Templates



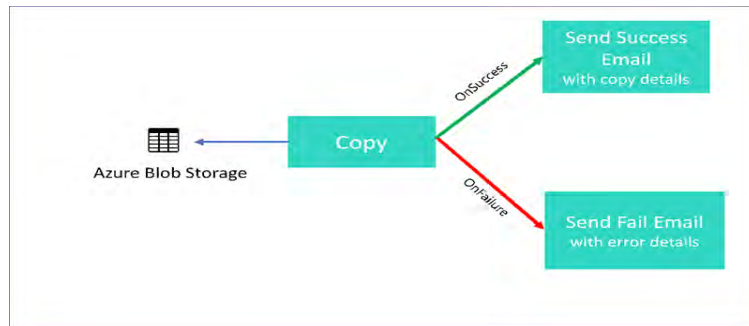
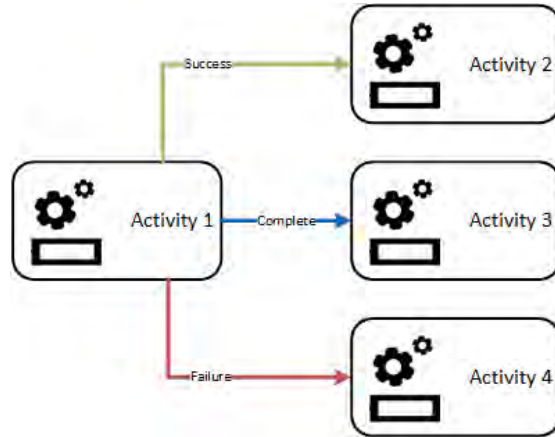
# Control Flow & Triggers

# ADFv2 Pipelines



# Activities

## Concepts



## Branching

Dependencies of activities in a pipeline

Possible constraints:

- On success
- On failure
- On completion

Also custom 'if' conditions will be available for branching based expressions

# Triggers

## How do pipelines get started

- on-demand
- Wall-clock Schedule
- Tumbling Window (aka time-slices in v1)
- Event on Blob Store

# Demo Control Flow

Get all data of a system by metadata

Pipeline check metadata



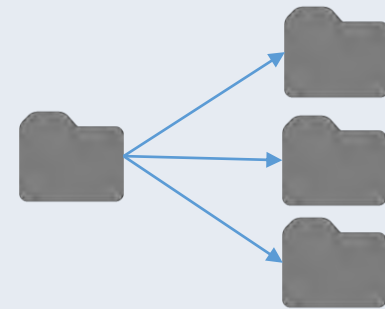
Activity: For each table  
in source

exec

exec

exec

Activity: Copy  
Data to Data Lake /  
Blob Store

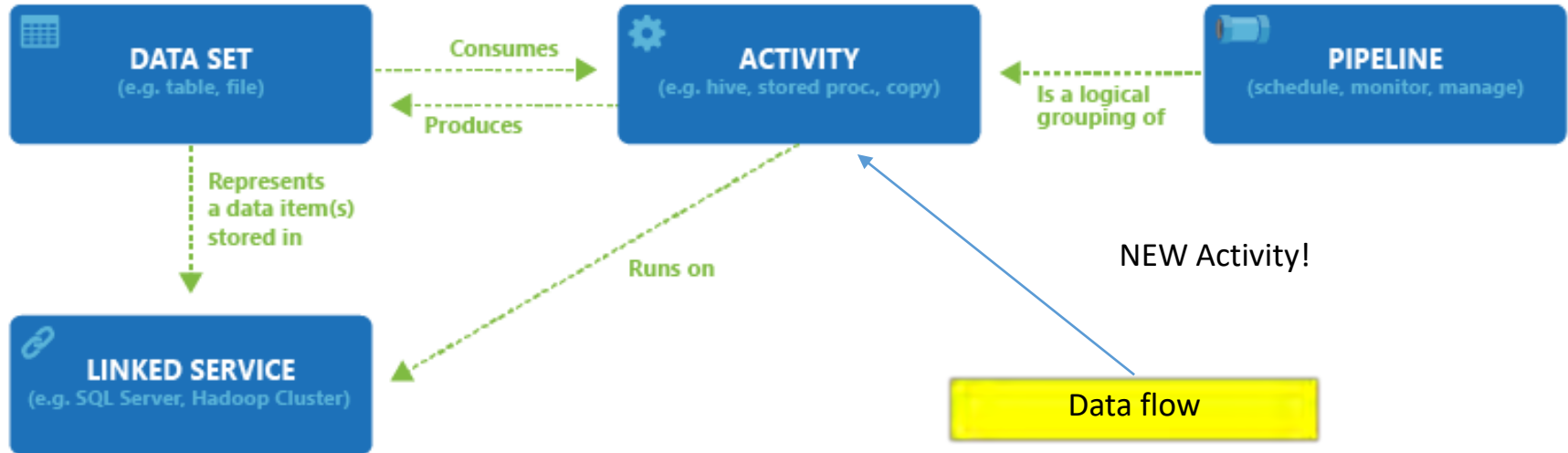


# Data Flow

Big data transformations without coding

# Data Factory Essentials

## Artefacts in Data Factory





# Visual / Mapping Data Flows



The screenshot displays a data flow tool interface with a central canvas and two side panels.

**Left Panel (Tools):**

- Search
- Multiple inputs/outputs
  - New Branch
  - Join
  - Conditional Split
  - Union
  - Lookup
- Schema modifier
  - Derived Column
  - Aggregate
  - Surrogate Key
- Row modifier
  - Exists
  - Select
  - Filter

**Central Canvas (Workflow):**

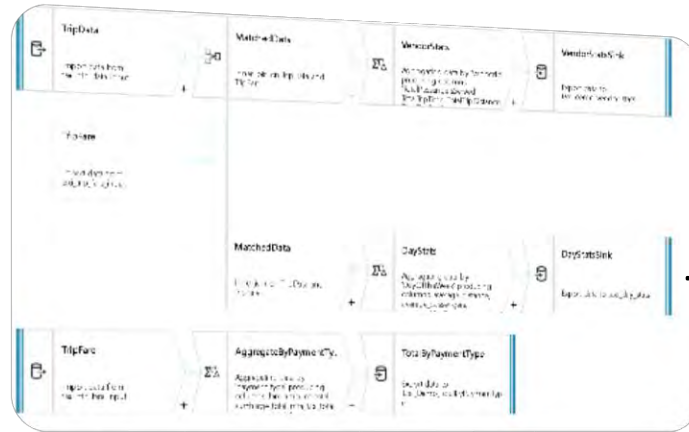
- sourceBatting** (Input data from BaseballBatting) connects to **CalcTotalBases** (23 total).
- sourcePlayer** (Input data from BaseballPlayer) connects to **PlayerInfoPlusOrigStats** (error join on sourcePlayer and BaseballBatting).
- CalcTotalBases** connects to **OrigSourceBatting** (Rejoining CalcTotalBases to OriginalBatting with column playerID and teamID, type: O, AG, N, H, R).
- PlayerInfoPlusOrigStats** connects to **createAggs** (Aggregating data by playerID, yearID including columns BA, CH, SH).
- OrigSourceBatting** connects to **createAggs**.
- createAggs** connects to **idn1** (error join on createAggs and PlayerInfoPlusOrigStats).
- idn1** connects to **sink1** (Sink data to MySQLDatabaseOutput).

**Right Panel (Visual Expression Builder):**

- Currently working on: **assener count**
- Filter:
- Categories: All, String, Math, Date, Logical, Input
- Functions list:
  - 122 **stdevSampleIf**( condition,  numeric value)
  - subDays**( date/timestamp,  days to subtract)
  - subMonths**( date/timestamp,  months to subtract)
  - substring**( string to subset,  from 1 based index,  number of characters)
  - 123 **sum**( numeric value)
  - 124 **sumDistinct**( numeric value)
  - 122 **sumDistinctIf**( condition,  numeric value)
  - 122 **sumIf**( condition,  numeric value)
  - 124 **tan**( numeric value)
  - 122 **tanh**( numeric value)
  - toBoolean**( string)
  - toDate**( string,  date format)

# Code-free Data Transformation At Scale

- Does not require understanding of Spark, Big Data Execution Engines, Clusters, Scala, Python ...
- Focus on building business logic and data transformation
  - Data cleansing
  - Aggregation
  - Data conversions
  - Data prep
  - Data exploration



... not ...

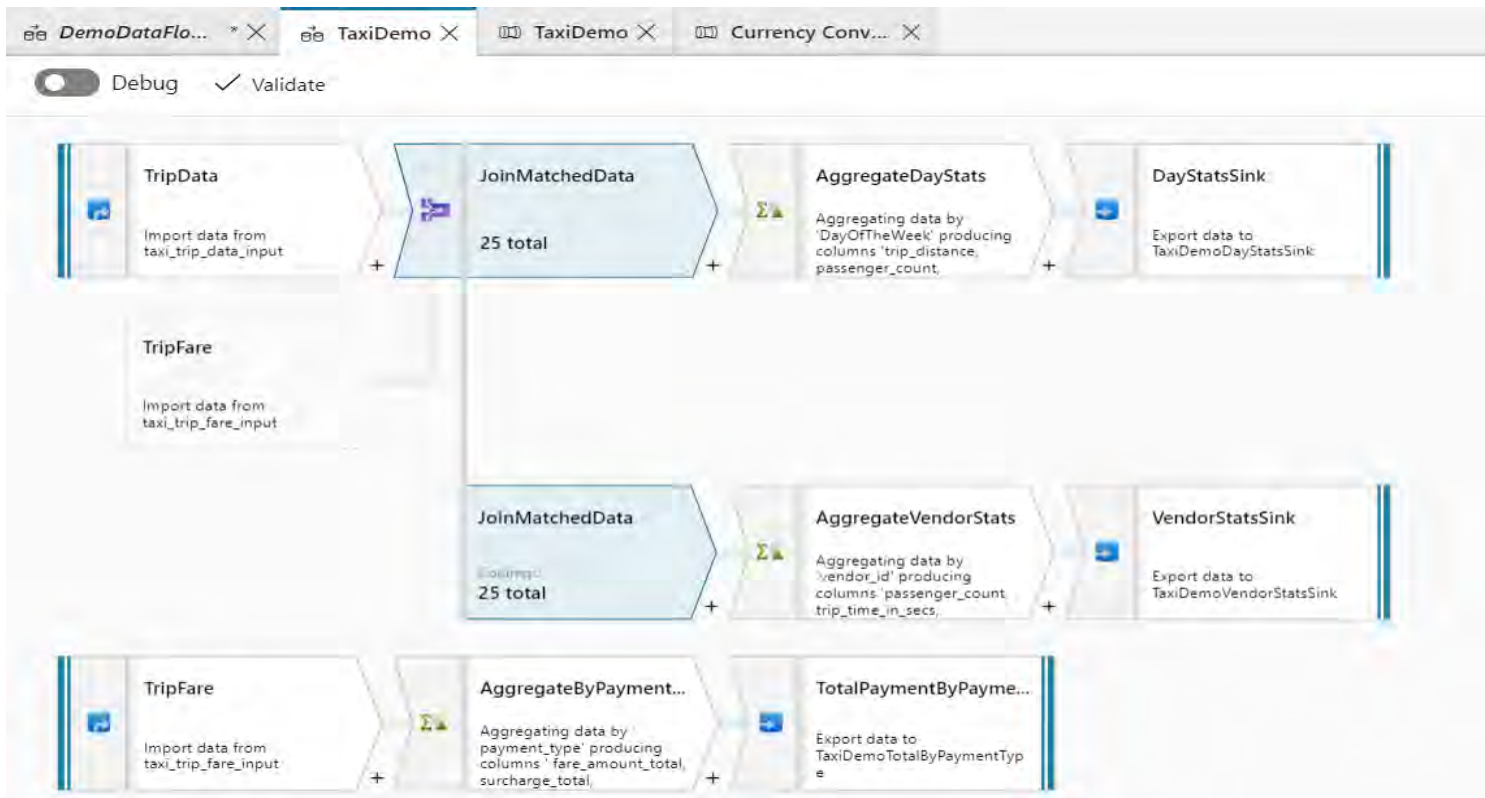
```
class SampleData {
  def tip: Double = 1.0
  def matched: Boolean = true
  def dayState: String = "DayState"
  def dayStateLink: String = "DayStateLink"
  def aggregatedByPaymentType: String = "AggregatedByPaymentType"
  def totalByPaymentType: String = "TotalByPaymentType"
}

def main(args: List[String]): Unit = {
  val sampleData = SampleData()
  val tip = sampleData.tip
  val matched = sampleData.matched
  val dayState = sampleData.dayState
  val dayStateLink = sampleData.dayStateLink
  val aggregatedByPaymentType = sampleData.aggregatedByPaymentType
  val totalByPaymentType = sampleData.totalByPaymentType

  // ... not ...
}
```

# Demo: Data Flow

## Visual Design of Transformations in Spark



# Visual Data Flow Key Tenets

- Visual “Data Flow Builder” / “Data Mapping”
- Extensible through scripting and expressions
- Data Flow can be embedded into ISV / SaaS apps
  - Embed UI
  - Embed Parameterize Data Flows
- A graphical UI for building data transformation routines on Spark
- Built for resiliency and operationalized environments

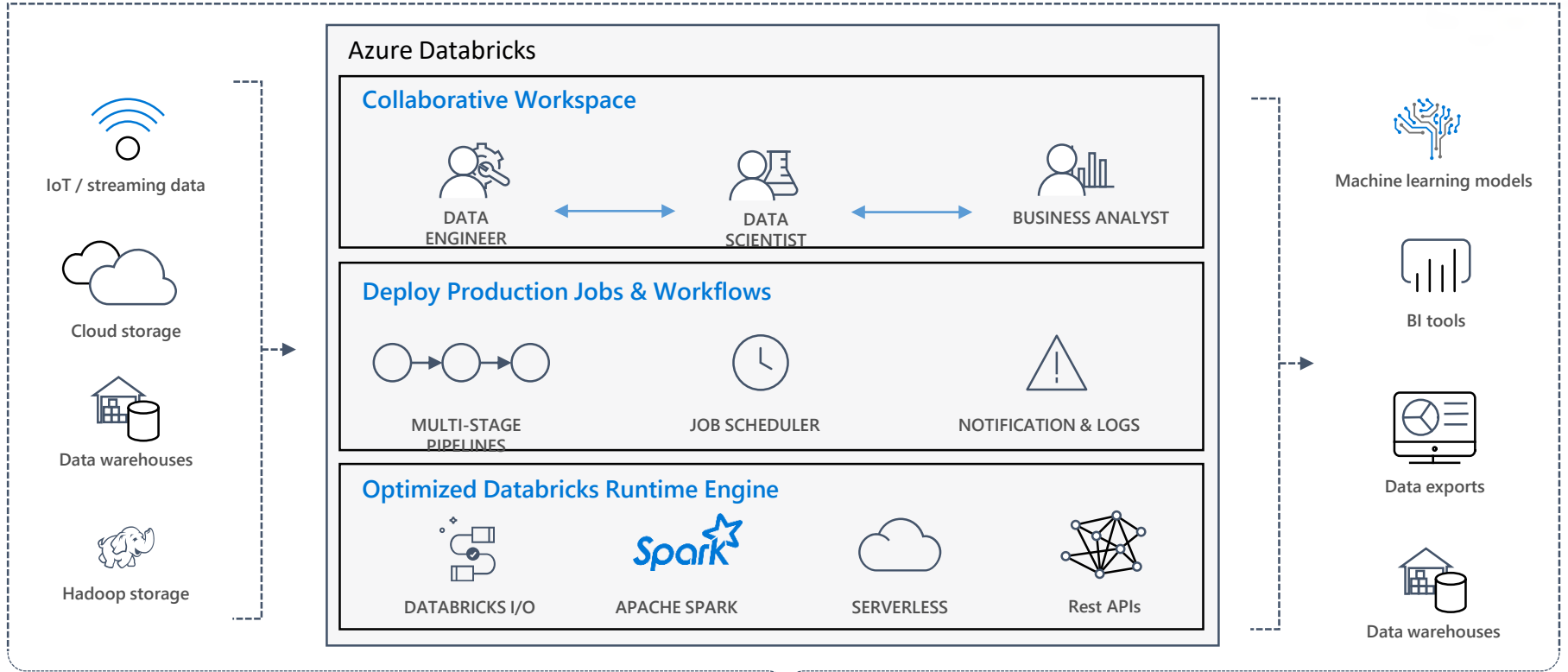


# Data flow started - what happens at databricks?

## Log4j output

```
21:39:35 INFO FileSourceScanExec: Pushed Filters:
18/12/03 21:39:35 INFO CodeGenerator: Code generated in 39.168945 ms
18/12/03 21:39:35 INFO MemoryStore: Block broadcast_34 stored as values in memory (estimate
18/12/03 21:39:35 INFO MemoryStore: Block broadcast_34_piece0 stored as bytes in memory (es
18/12/03 21:39:35 INFO BlockManagerInfo: Added broadcast_34_piece0 in memory on 10.139.64.5
18/12/03 21:39:35 INFO SparkContext: Created broadcast 34 from load at External.scala:73
18/12/03 21:39:35 INFO FileSourceScanExec: Planning scan with bin packing, max size: 419430
18/12/03 21:39:35 INFO SparkContext: Starting job: load at External.scala:73
18/12/03 21:39:35 INFO DAGScheduler: Got job 21 (load at External.scala:73) with 1 output p
18/12/03 21:39:35 INFO DAGScheduler: Final stage: ResultStage 35 (load at External.scala:73
18/12/03 21:39:35 INFO DAGScheduler: Parents of final stage: List()
18/12/03 21:39:35 INFO DAGScheduler: Missing parents: List()
18/12/03 21:39:35 INFO DAGScheduler: Submitting ResultStage 35 (MapPartitionsRDD[124] at lo
18/12/03 21:39:35 INFO MemoryStore: Block broadcast_35 stored as values in memory (estimate
```

# Overview Azure Databricks



Enhance Productivity

Build on secure & trusted cloud

Scale without limits

# Databricks Backend- Configuration

● adfstefankirnerfivepure



Edit

Clone

Restart



Configuration

Notebooks (0)

Libraries (10)

Event Log

Spark UI

Driver Logs

Spark Cl

## Cluster Mode

High Concurrency

Optimized to run concurrent SQL, Python, and R workloads.  
Does not support Scala. Previously known as Serverless.

Standard

Recommended for single-user clusters and Scala workloads.

## Databricks Runtime Version

5.0 (includes Apache Spark 2.4.0, Scala 2.11)

## Python Version

2

## Driver Type

Standard\_F4s

8.0 GB Memory, 4 Cores, 0.5 DBU

## Worker Type

Standard\_F4s

8.0 GB Memory, 4 Cores, 0.5 DBU

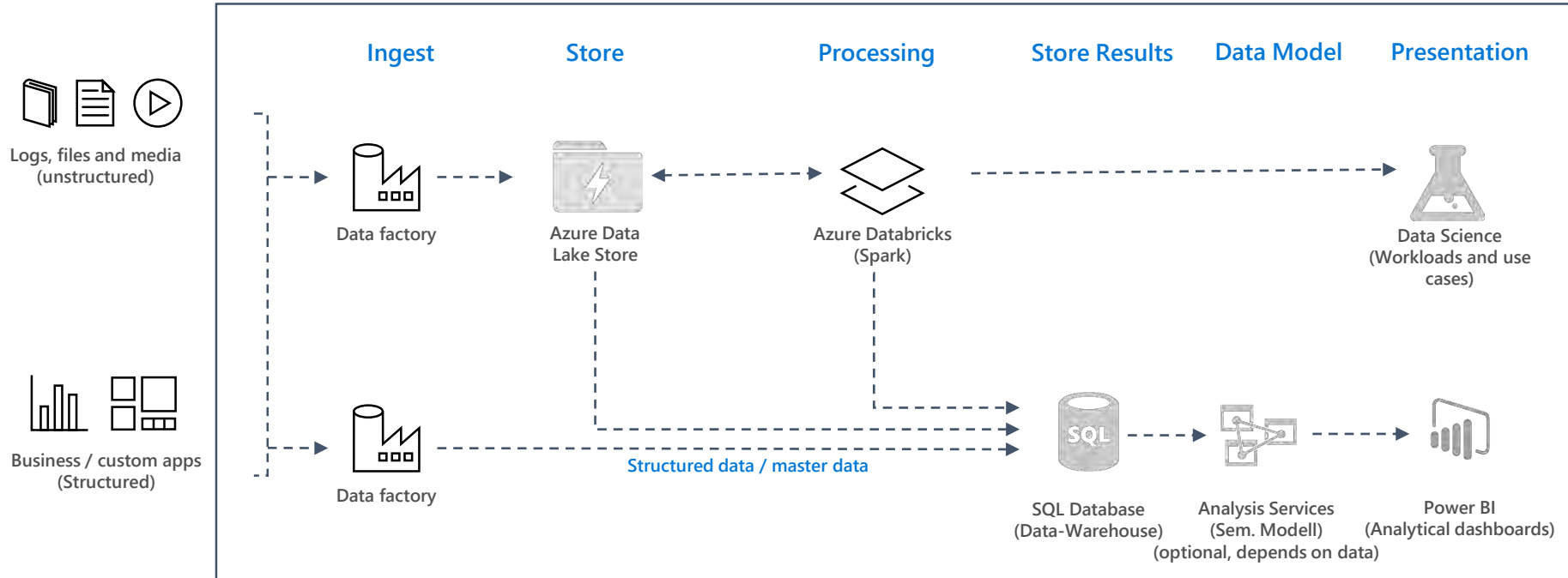
## Workers

1

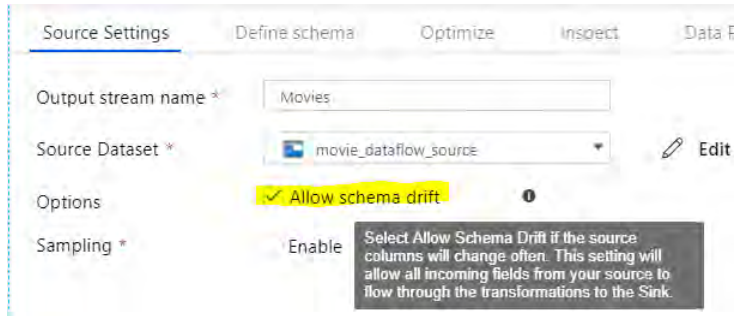
Enable autoscaling 



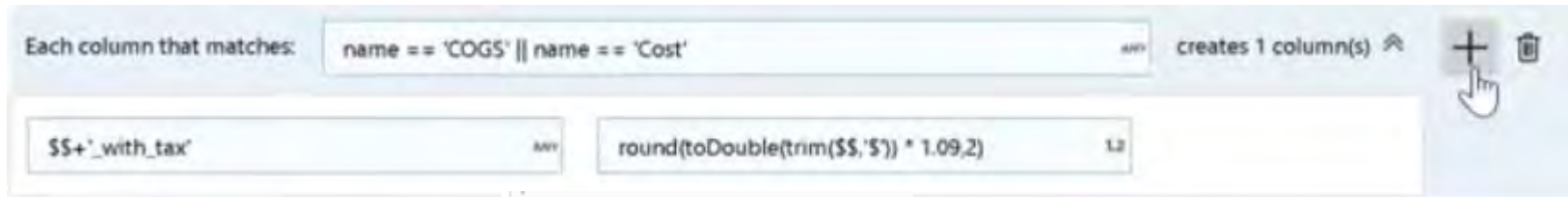
# In context of Microsofts “Modern DWH”



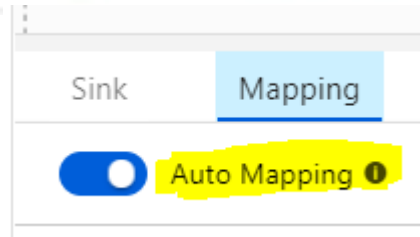
# Nice Feature: Handling Schema Changes



Data flow will accept both columns (here in derived columns)



Use Auto Mapping  
in Sink



# Conclusion: Data Flow in ADF

Is cool because...

- **visual design** for fast learning & understanding
- ETL using spark technology in the background which **could** scale (but does not have to in any case)
- Azure Databricks as **elastic processing engine** for different workloads, tools and user groups
- Integration in Control Flow enables modelling **dependencies** and **cost-efficient** orchestration of Azure resources

# Data Flow Limited Preview Support & SLAs

- Azure SLAs are NA for preview services (private or public preview) until GA of the service.
- Limited Preview Support
  - Handled directly with the Azure Engineering team via [adfdataflowext@microsoft.com](mailto:adfdataflowext@microsoft.com). Turn-around time on fixing issues during private preview will depend upon access to customer data sources and customer Databricks clusters for RCA and debugging.
- Public Preview Support
  - Normal Azure customer service channels

# February Release

## Data Flows

- DFs included in general V2 version
- The new Templates also support DF
- On-demand clusters managed by ADF instead of BYO
- Move or delete source files after execution (!)
- Auto-detect data types & wildcards
- Exec data flow: parameters & cluster size
- Still whitelisting of subscription needed

Q+A

# Links and further informations

1. Microsoft documentation:

<https://docs.microsoft.com/en-us/azure/data-factory/>

2. Azure Data Factory – data flows preview documentation

<https://github.com/kromerm/adfdataflowdocs>

3. Cool screencasts about data flows

<https://github.com/kromerm/adfdataflowdocs/tree/master/videos>

4. Another good blogpost about ADF Data Flows

<https://visualbi.com/blogs/microsoft/azure/azure-data-factory-data-flow-activity/>

5. Comparison ADF Data Flows vs. SSIS vs. T-SQL

<https://sqlplayer.net/2018/12/azure-data-factory-v2-and-its-available-components-in-data-flows/>



Thank you very much for your attention.  
Vielen Dank für Eure Aufmerksamkeit.



**Unbedingt! Euer Feedback zu meiner Session**

