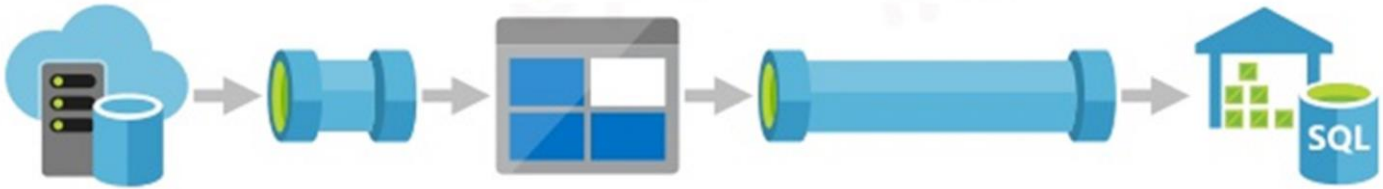

Azure Data Factory
Cloud version of SSIS

What can you do in Azure Data Factory?

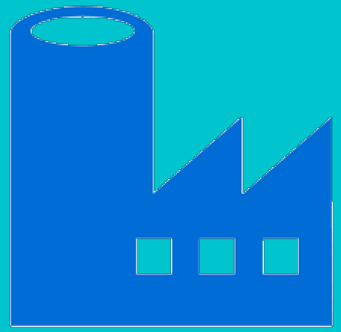


Copy Data

More than 80 connectors to different services are available

Transform Data

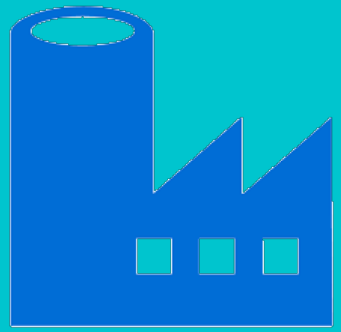
Using newly added Data Flow, now Data Factory is complete cloud based ETL tool.



Azure Data Factory

Definition:

Azure Data Factory (ADF) is a hybrid data integration service that enables you to quickly and efficiently create automated data pipelines – without having to write any code!



Azure Data Factory

- Hybrid Data Integration Service
- Simplifies ETL at scale
- Enables modern data integration
- Drag and drop interface
- Over 80 connectors available
- Move, transform and save data
- Managed Service
- Create Data Driver workflows
- Orchestrate and automate data movement
- Transform and store data
- Operationalize the process
- ETL or ELT scenarious

Data Factory on Azure Ecosystem

01 Migration?

Data Factory excels in periodic data loads and transformation instead.

02 Streaming?

ADF can orchestrate, but there are other dedicated services for streaming

03 Transformations?

Data flows for simple ones, but you can use Databricks or HDInsight for more complex transforms

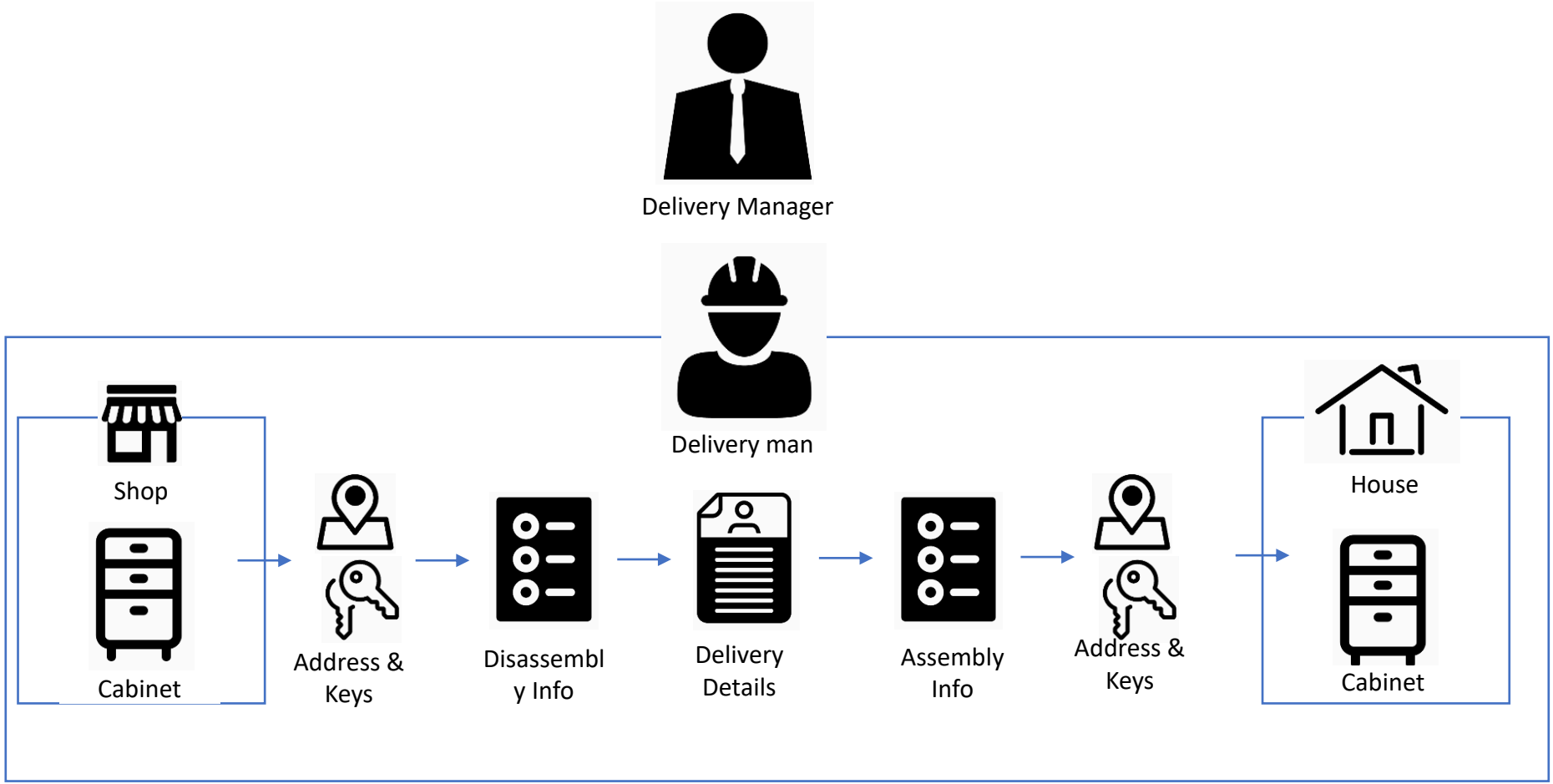
SSIS vs Data Factory

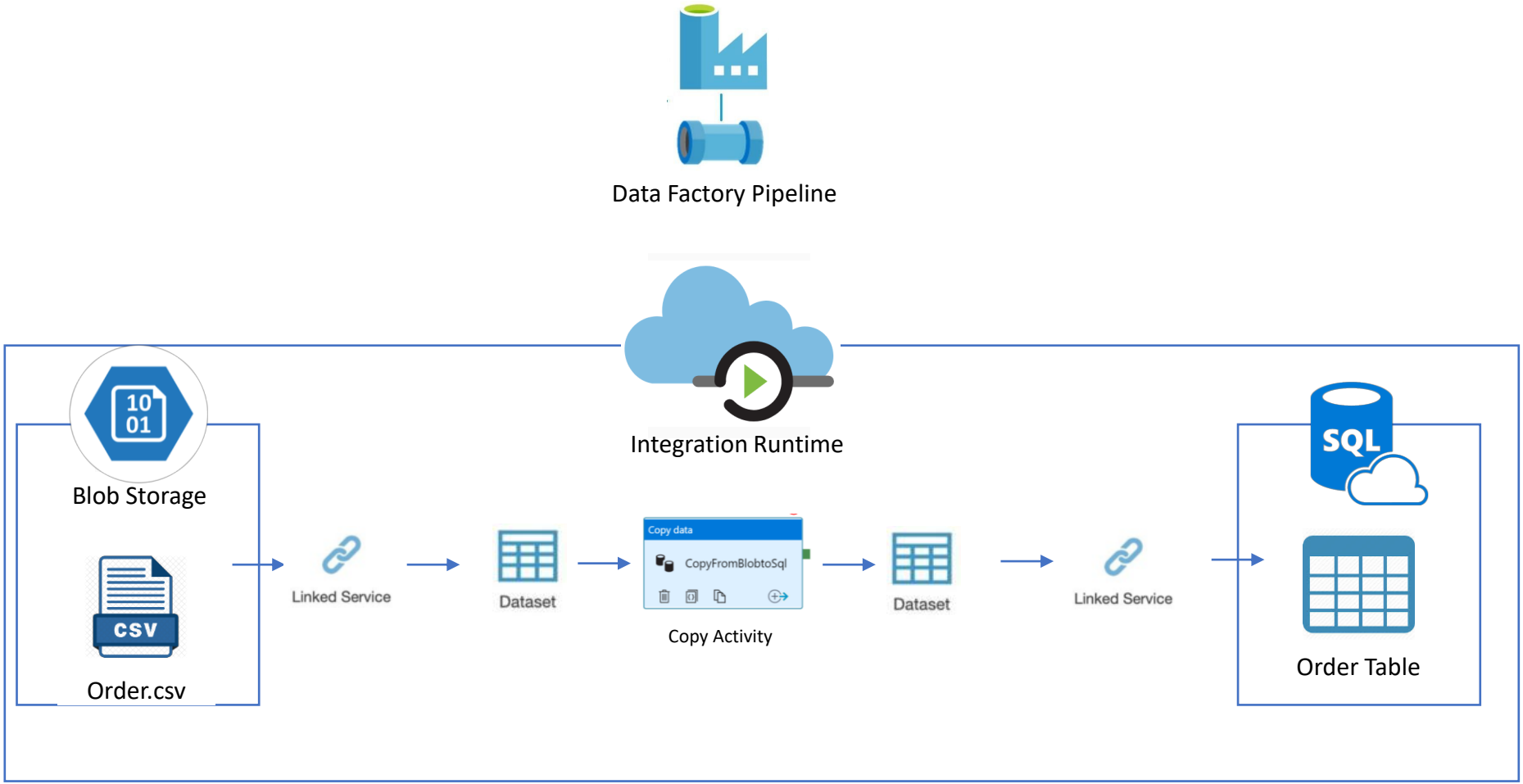
SSIS	Data Factory
More code-free transformations	Much higher scalability
On Premises connectors (e.g excel)	Cloud and SaaS Connectors
	Event based Triggers
	Can use SSIS Packages

Data Factory considerations

Two versions	Build options	Highly integrated	No data storage	Security standards
ADF V2 is the current and improved version	PowerShell, .Net, Python, REST, ARM	DevOps, Key Vault, Monitor, Automation	Need to persist data by the end.	HTTP/TLS whenever possible

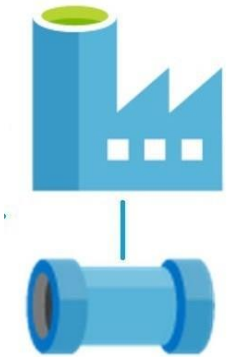
Azure Data Factory Components


Delivery Manager
Delivery man
Shop
Cabinet
Address &
Keys
Disassembl
y Info
Delivery
Details
Assembly
Info
Address &
Keys
House
Cabinet


Data Factory Pipeline
Integration Runtime
Blob Storage
Order.csv
Linked Service
Dataset
Copy data
CopyFromBlobtoSql
Copy Activity
Dataset
Linked Service
Order Table

Data Factory vs SSIS

Azure Data Factory	SSIS
Pipeline	Package
Linked Service	Connection manager
Source	Source
Sink	Destination
Activity	Control flow task
Data Flow	Data flow



Data Factory
Pipeline

- Data Factories can contain one ore more pipelines
- Logical group of Activities
- Manage Activities as a set
- One Pipeline can have one or more activities

Azure Data Factory Activities

- Represents a processing step in the pipelines
- Actions to perform on data
  - Ingest data
  - Transform data
  - Store data
- Can be linked
  - Execute sequentially or
  - Run in parallel

Activity types

01 Data movement activities

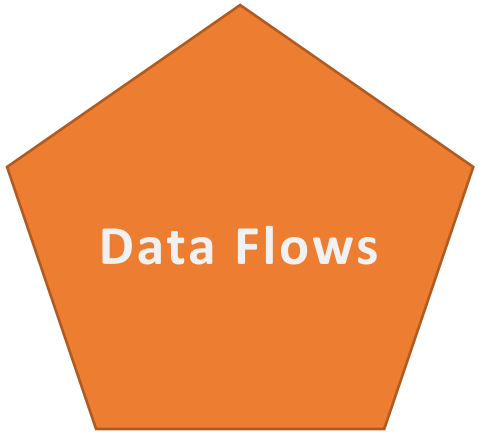
Copy data amongst data stores located on-premises and in the cloud
Data stores – Blob storage, Cosmos DB, Amazon Redshift, Google BigQuery Hive, Maria DB…etc.

02 Data transformation activities

Transform and enrich data
e.g. Hive, Pig, MapReduce, Spark or Databricks

03 Control activities

Control pipeline flow
e.g. ForEach, Web



- Data Flow is a new feature of Azure Data Factory (ADF) that allows you to develop graphical data transformation logic that can be executed as activities within ADF pipelines.
- Two types:
  - Mapping
  - Wrangling

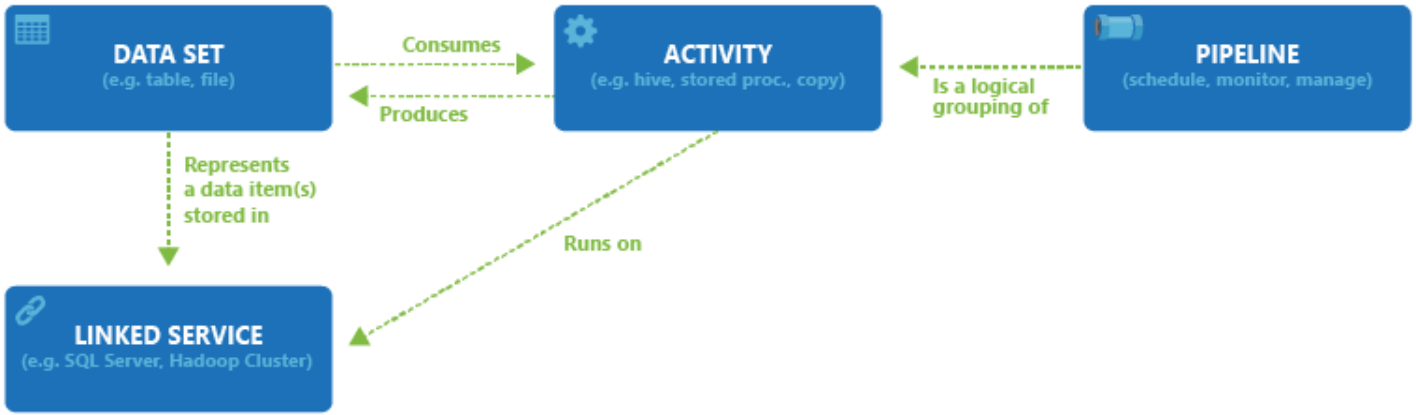
Dataset

- Simply point or reference the data
- Reference data used in an Activity
  - Files
  - Folders
  - Documents
  - Tables

Linked service

- Similar to connection string
- Represent the connection information to connect to external resources
  - Datastores like Azure SQL Server
  - Compute resource e.g. Spark Cluster

ADF Components



Integration Runtimes

- Provides fully managed, serverless compute infrastructure
  - You don't have to worry about infrastructure provision, software installation, patching, or capacity scaling.
  - Pay only for duration of actual use
- Bridges between the activity and linked service
  - Activity defines the action
  - Linked service define the location

Integration Runtimes

- **Data Integration Capabilities**
  - **Data Flow**
  - **Data Movement**
    - Format conversion, column mapping, serialization/ deserialization etc.
    - Provides the native compute to move data between cloud data stores in a secure, reliable, and high-performance manner.
  - **Activity dispatch** (e.g. Databricks Notebook, HDInsight Hive, pig, spark activity, SP, ADL Analytics U-SQL activity)
  - **SSIS Package execution**

Integration Runtimes

Specify the infrastructure to run activities

Azure Integration Runtime

Work on public networks
Responsible for data flows, data movements, and activity dispatches

Self-hosted Integration Runtime

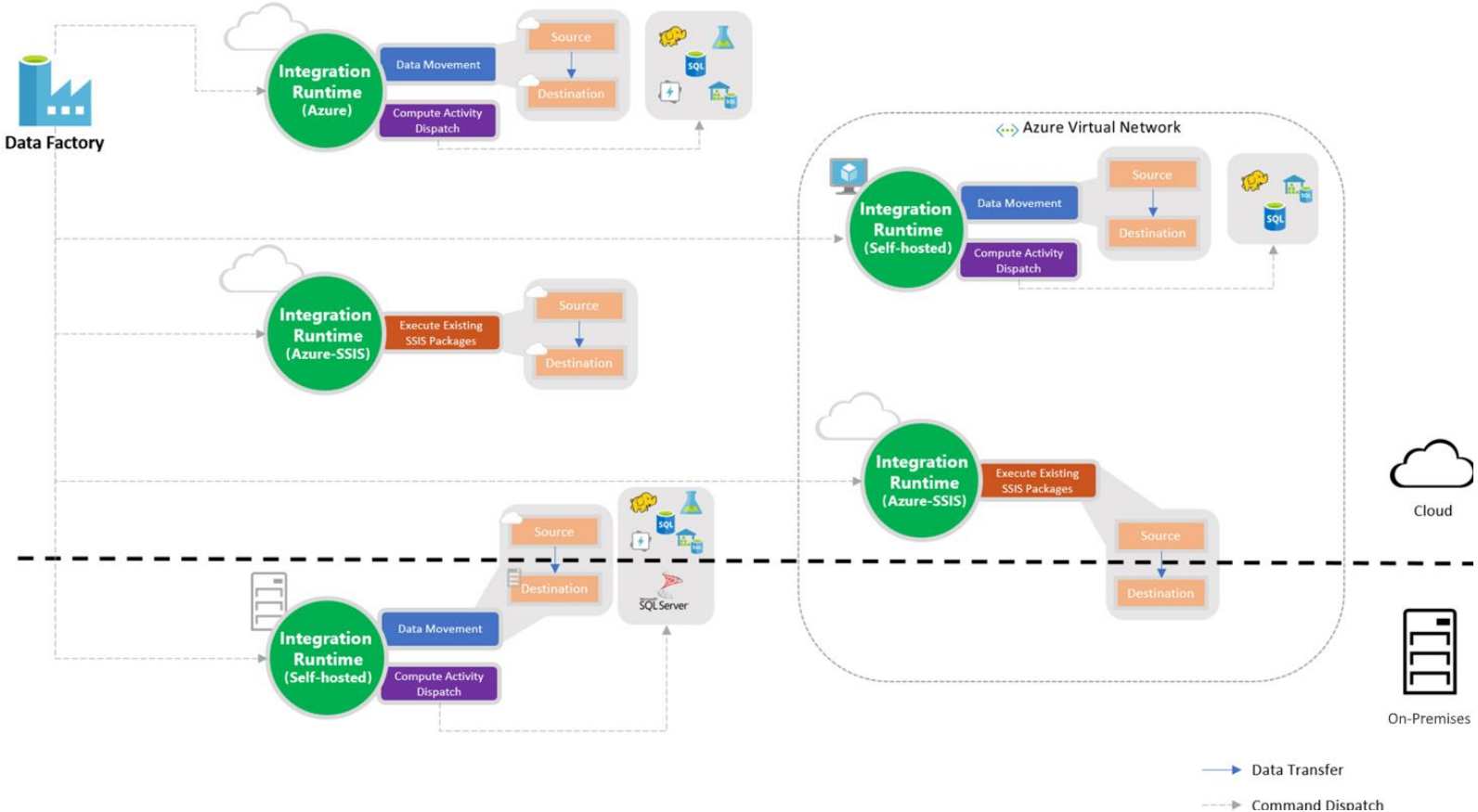
Work on public and private networks
Provide data movement and activity dispatch capabilities
Need to install on on-premises machine or a virtual machine inside private network

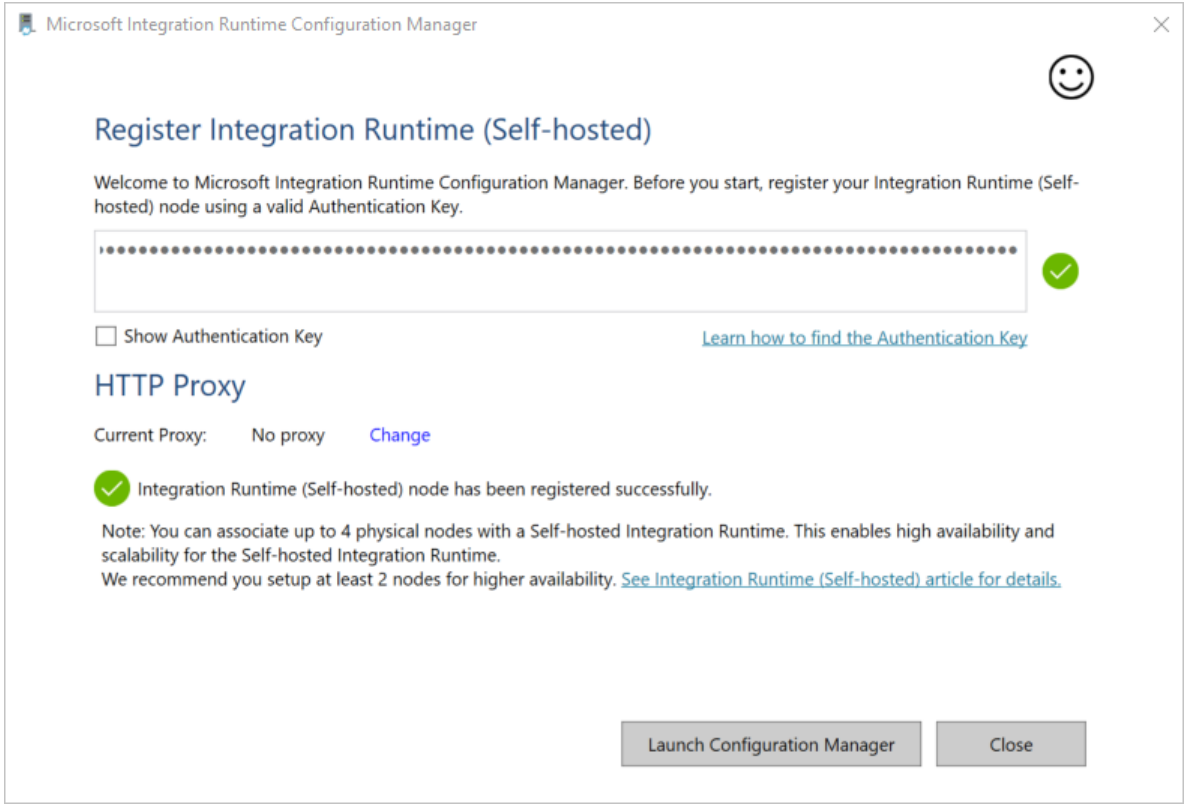
SSIS Integration Runtime

Supports SSIS package execution
Works on public and private networks

Integration Runtimes

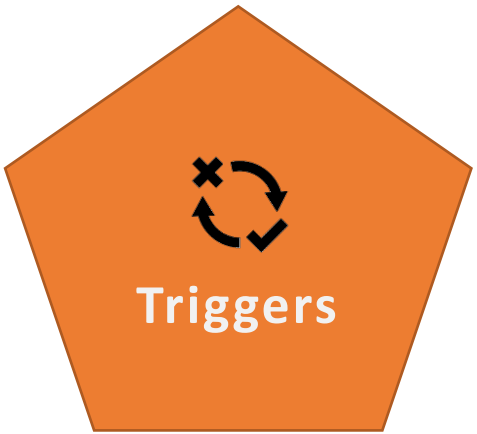
IR type	Public network	Private network
Azure	Data Flow Data movement Activity dispatch	
Self-hosted	Data movement Activity dispatch	Data movement Activity dispatch
Azure-SSIS	SSIS package execution	SSIS package execution


Data Factory
Integration Runtime (Azure)
Data Movement
Compute Activity Dispatch
Source
Destination
Azure Virtual Network
Integration Runtime (Self-hosted)
Data Movement
Compute Activity Dispatch
Source
Destination
Integration Runtime (Azure-SSIS)
Execute Existing SSIS Packages
Source
Destination
Integration Runtime (Azure-SSIS)
Execute Existing SSIS Packages
Source
Destination
Cloud
Integration Runtime (Self-hosted)
Data Movement
Compute Activity Dispatch
Source
Destination
SQL Server
On-Premises
Data Transfer
Command Dispatch


Microsoft Integration Runtime Configuration Manager
Register Integration Runtime (Self-hosted)
Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.
Show Authentication Key
Learn how to find the Authentication Key
HTTP Proxy
Current Proxy: No proxy Change
Integration Runtime (Self-hosted) node has been registered successfully.
Note: You can associate up to 4 physical nodes with a Self-hosted Integration Runtime. This enables high availability and scalability for the Self-hosted Integration Runtime.
We recommend you setup at least 2 nodes for higher availability. See Integration Runtime (Self-hosted) article for details.
Launch Configuration Manager
Close

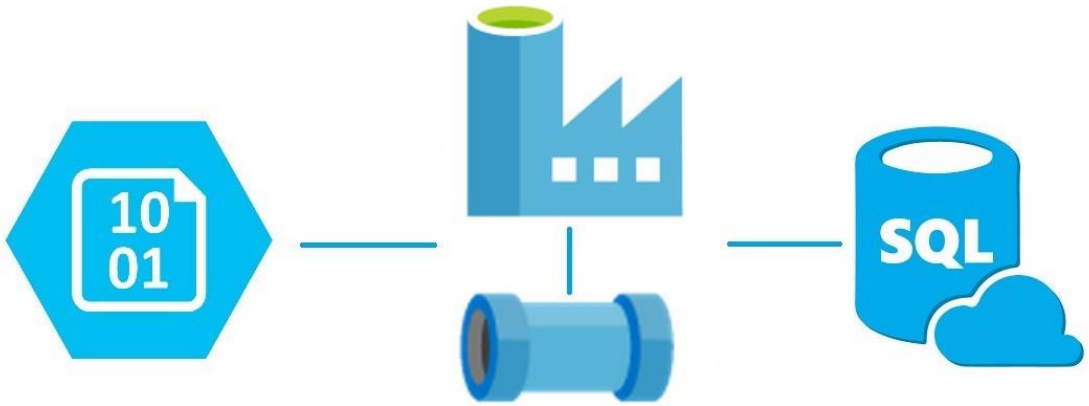
Integration Runtimes

- Default IR – AutoResolveIntegrationRuntime
- Create Azure IR
  - When you want to explicitly define the location of IR
  - Virtually group the activities executions on different IR for management purpose

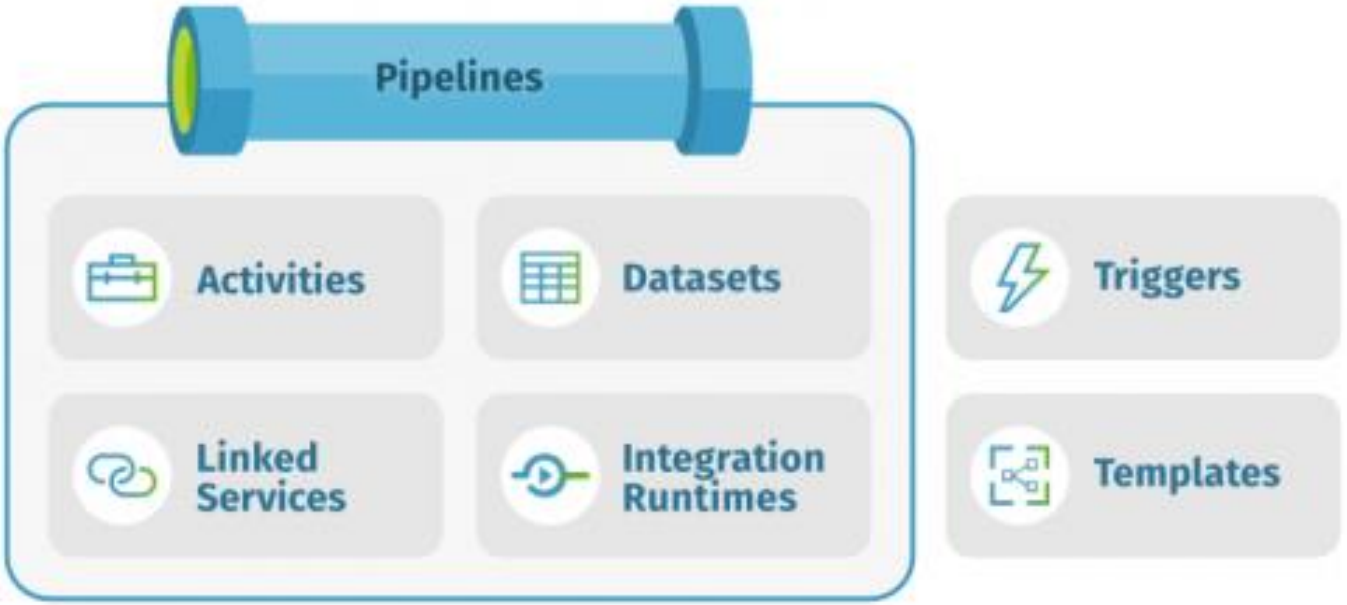


- **Execute pipeline**
- **Many to many relationship b/w pipeline and trigger**
- **Three types of Trigger**
  - **Schedule Trigger** – Invoke pipeline on a wall-clock schedule
  - **Tumbling Window Trigger –** Operates on a periodic interval, also retain state
  - one-to-one relationship
  - Advance configuration options - Dependencies, delay, retry, concurrency
  - Properties - trigger().outputs.WindowStartTime/WindowEndTime
  - **Event-based Trigger –** trigger pipeline in response to an event
    - e.g. Arrival/deletion of file in Blob storage
    - Event trigger with Azure Event Grid Service
    - Properties – triggerBody().folderPath/fileName

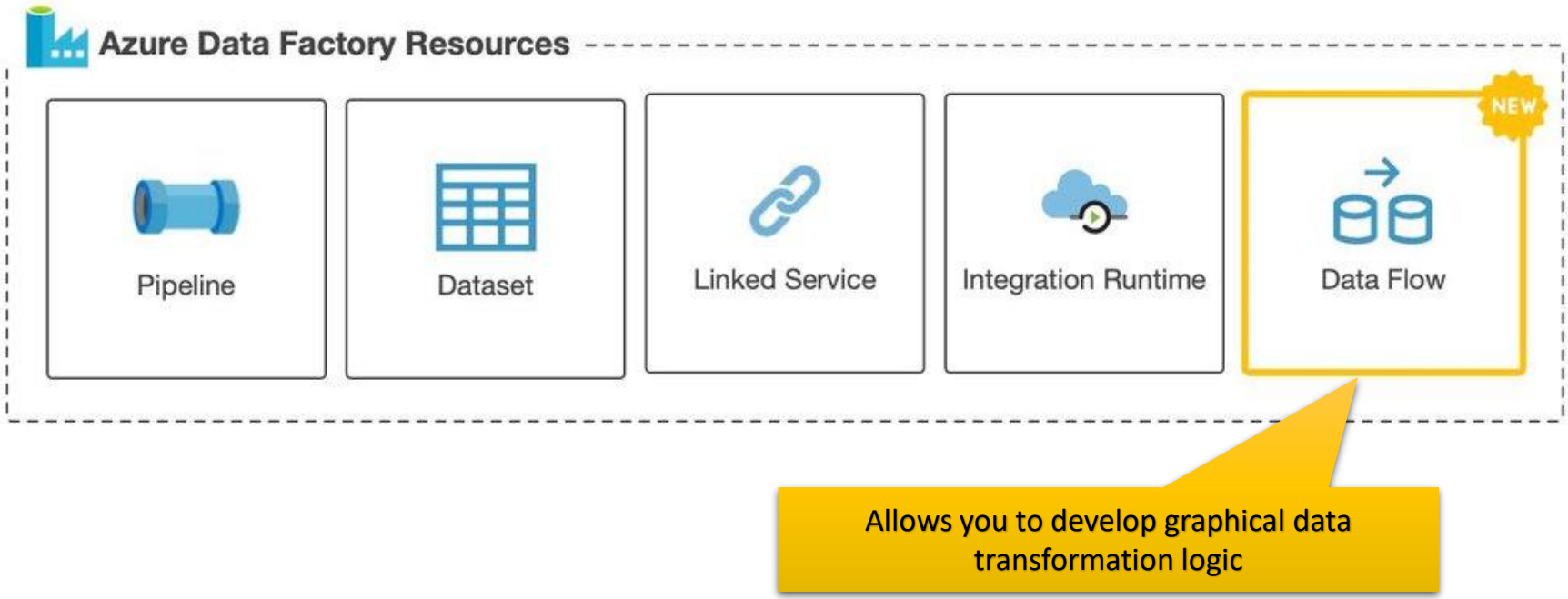
Demo: Copy Activity



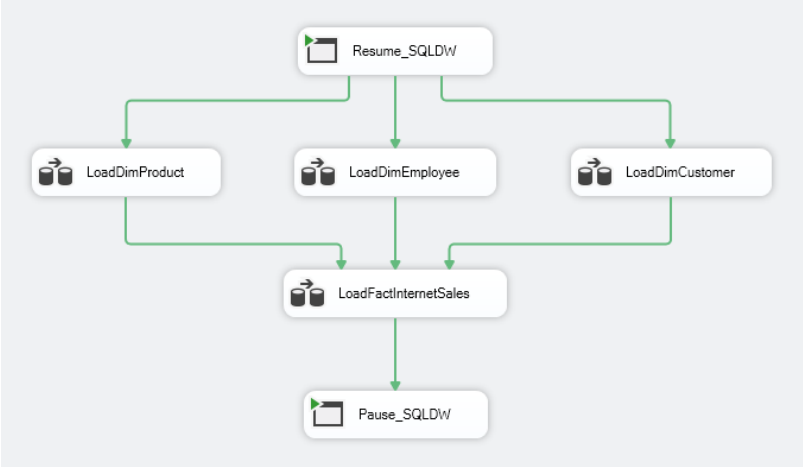
Summary


Pipelines
Activities
Datasets
Triggers
Linked Services
Integration Runtimes
Templates

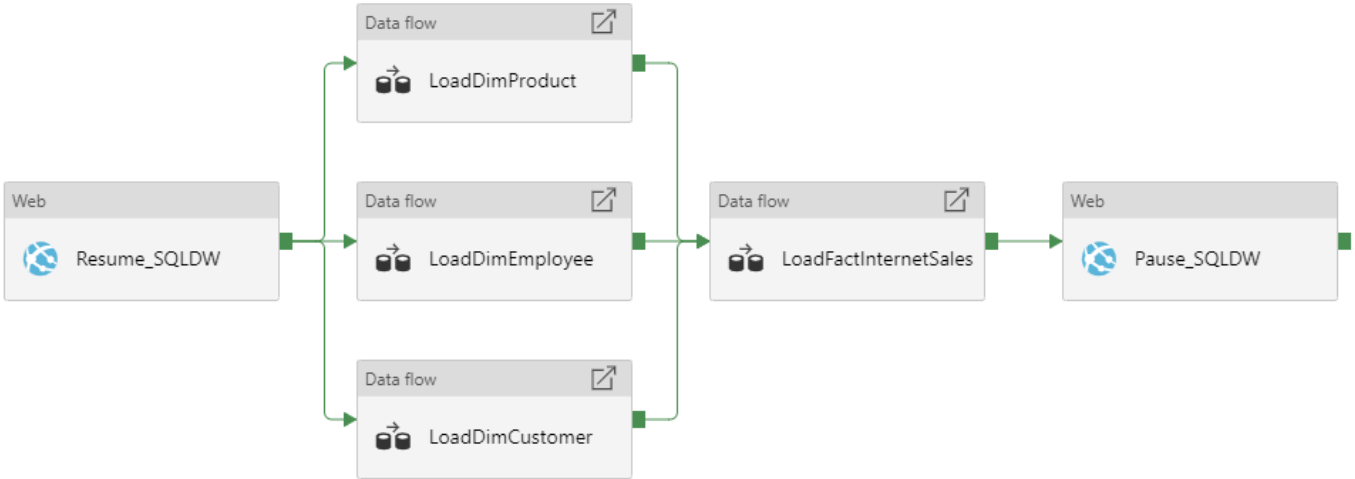
Data Flows


Azure Data Factory Resources
Pipeline
Dataset
Linked Service
Integration Runtime
NEW
Data Flow
Allows you to develop graphical data transformation logic

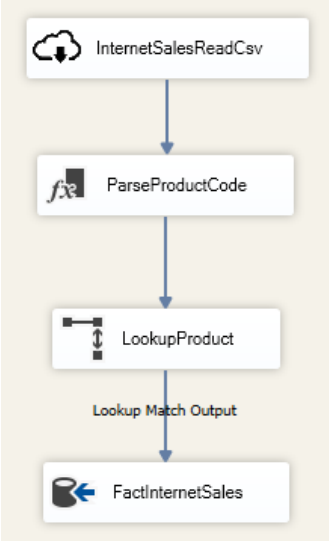
Example of the SSIS Control Flow tab for loading our data mart tables:



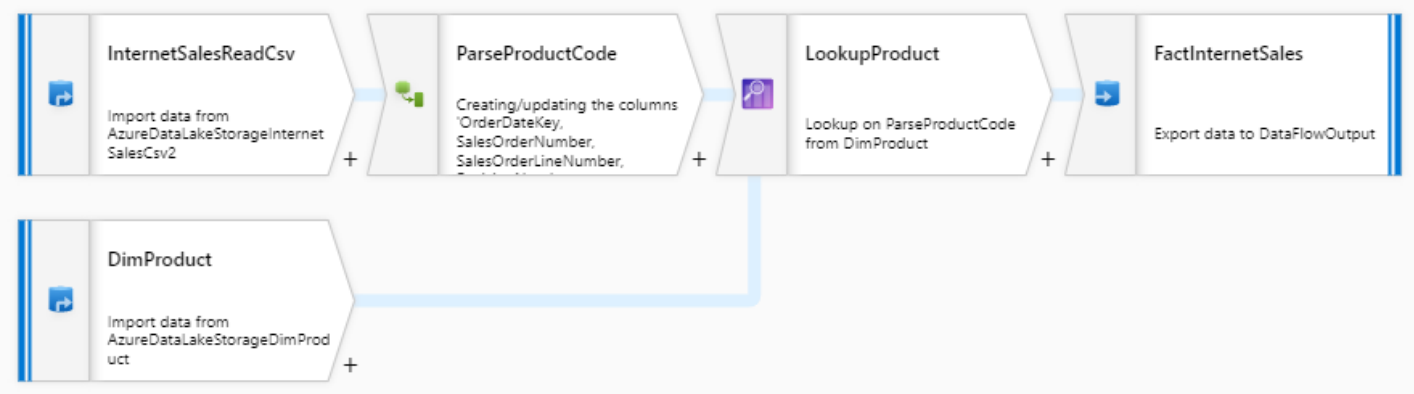
Example of the ADF Pipeline for loading our data mart tables:

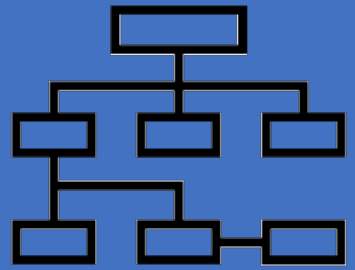


Example of SSIS Data Flow tab for loading the FactInternetSales table:



Example of ADF Mapping Data Flows for loading the FactInternetSales table:



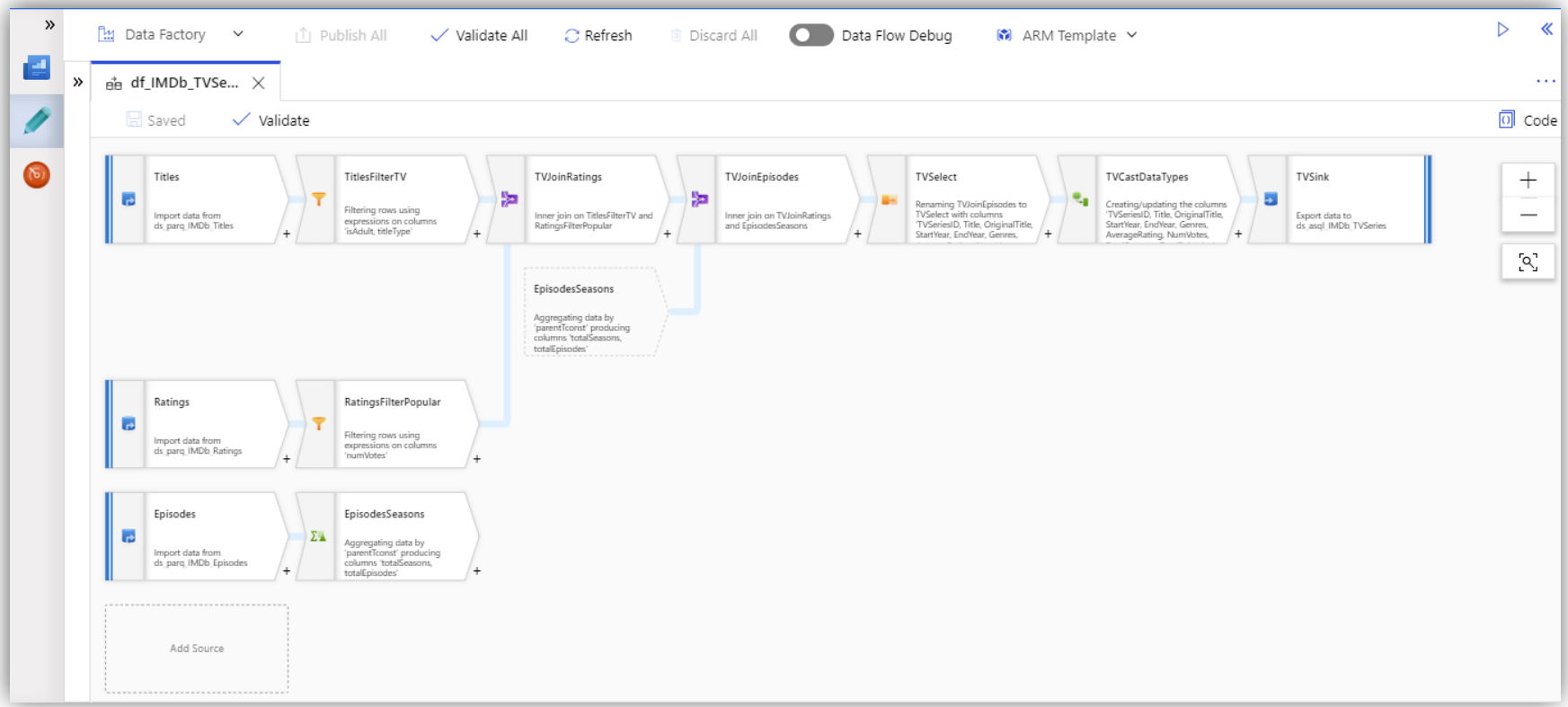


Data Flow

Mapping Data flow – Transform Data (Known data and schema)

Wrangling Data flow – Prepare and explore data using power query (known or unknown datasets)

Mapping Data Flows



Mapping Data Flow Actions

Multiple Inputs/outputs

- Join
- Conditional Split
- Exists
- Union
- Lookup

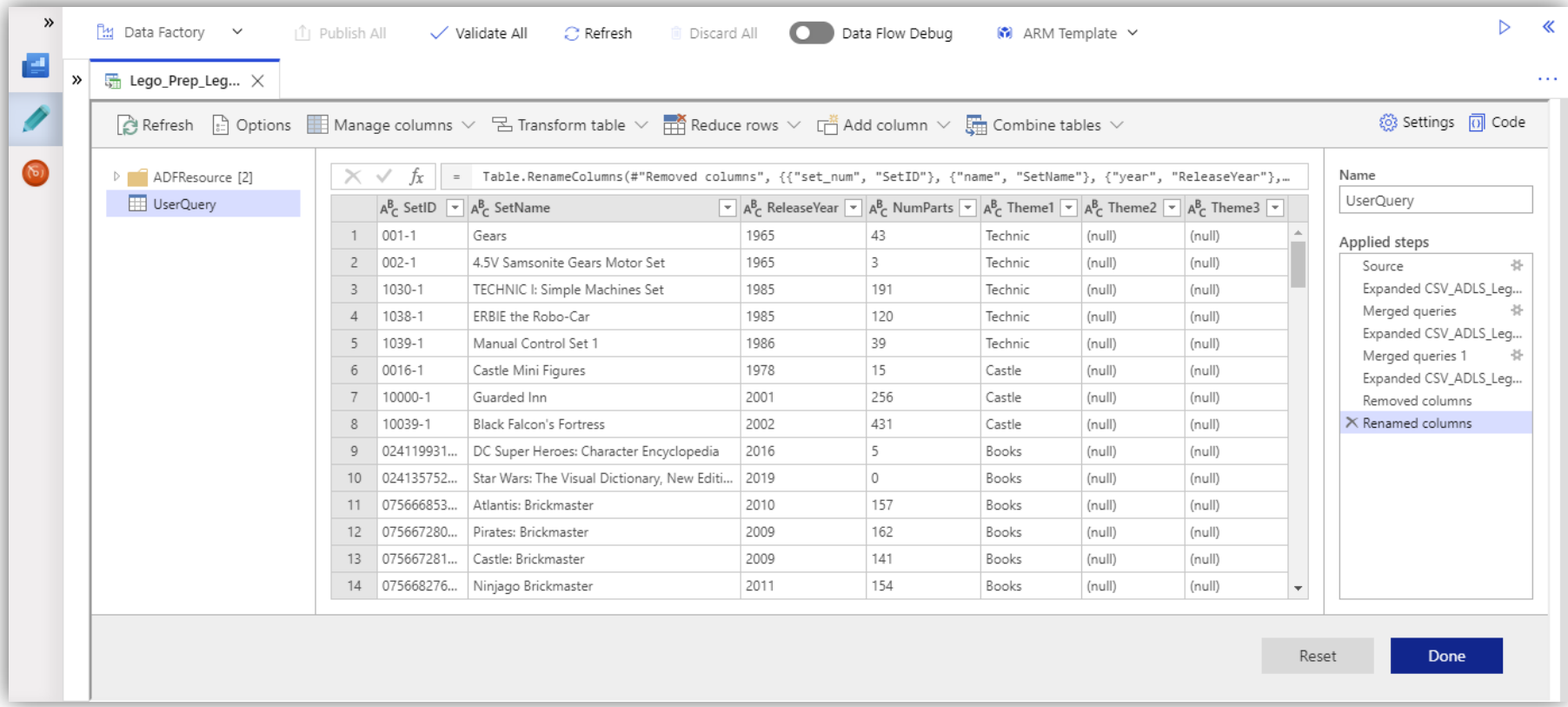
Schema Modifiers

- Derived Columns
- Select
- Aggregate
- Surrogate key
- Pivot
- Unpivot
- Window

Row Modifiers

- Filter
- Sort
- Alter Row

Wrangling Data Flows



Data flows behind the scene

- Behind the scene Data flow will execute on Azure Databricks using Spark
- ADF internally handles all the code translation, spark optimization and execution of transformation

Spark

Basics

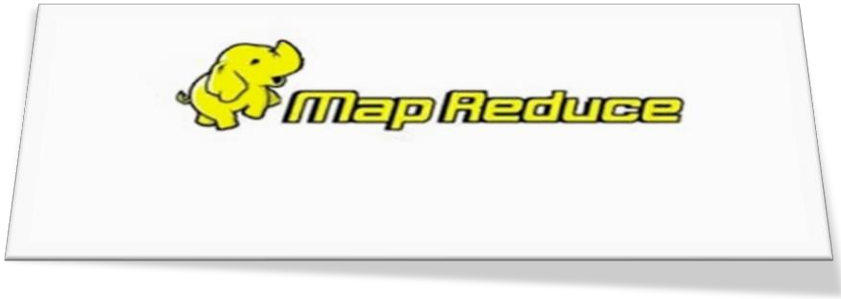
Why Map reduce introduced

Tradition system were failing

Centralized server to store and process data which creates bottleneck

Google introduced Map Reduce

Divide the task into small parts and assign them to many computers



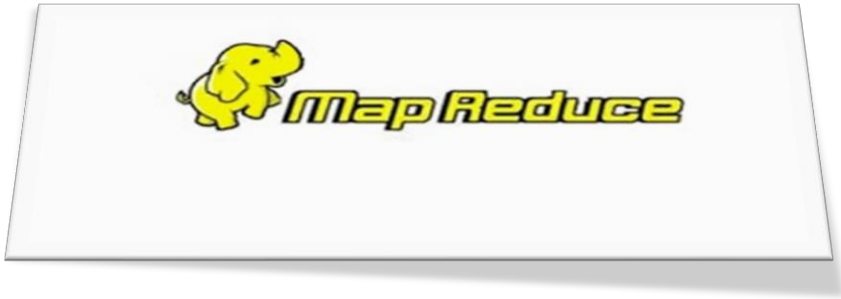
Challenges with Map Reduce

Data-sharing abstraction

Concurrent data access to memory across the cluster

Inefficient use of resources

Poor memory utilization by spilling to disk after each job



Apache Spark

Big Data Tool

“Spark is an open source unified analytics engine for large-scale data processing.”

1

Fast and efficient

Supports structured, semi-structured or unstructured.

2

In-memory engine

100 times faster than Hadoop

3

Highly scalable architecture

Process terabytes of data in parallel

4

Unifies variety of use cases

Batch processing, streaming ingestion, machine learning and advanced analytics

Hadoop

HDFS

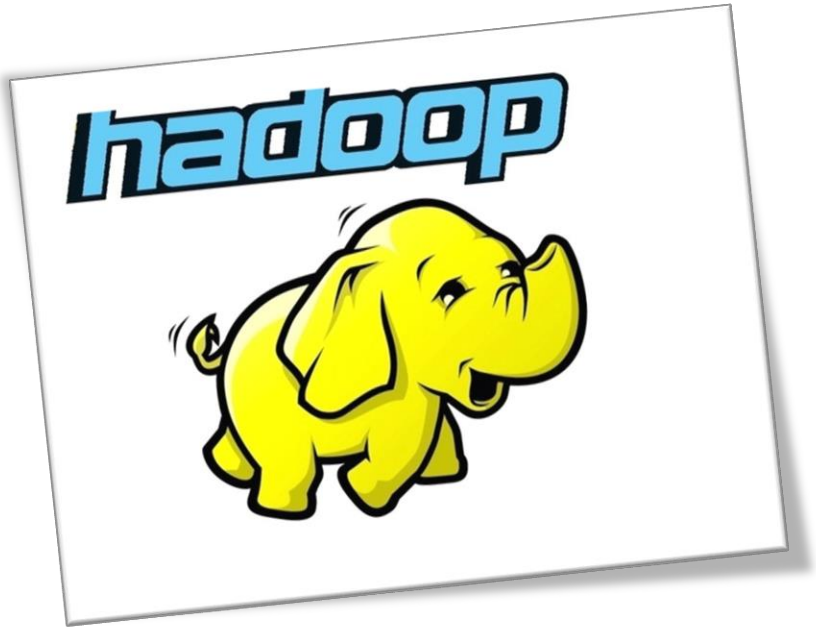
A file system to manage the storage of data

MapReduce

A framework to define a data processing task

YARN

A framework to run the data processing task



Co-ordination between Hadoop Blocks

1. Step One

User defines map and reduce tasks using the MapReduce API

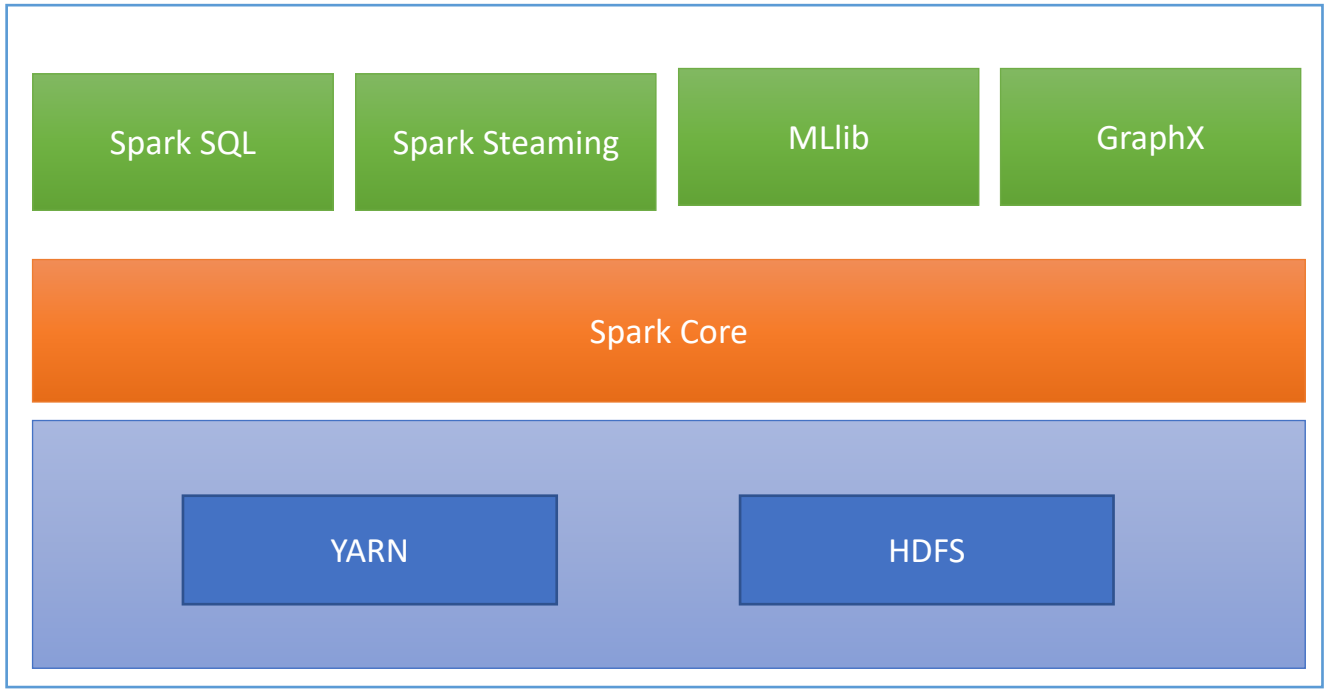
2. Step Two

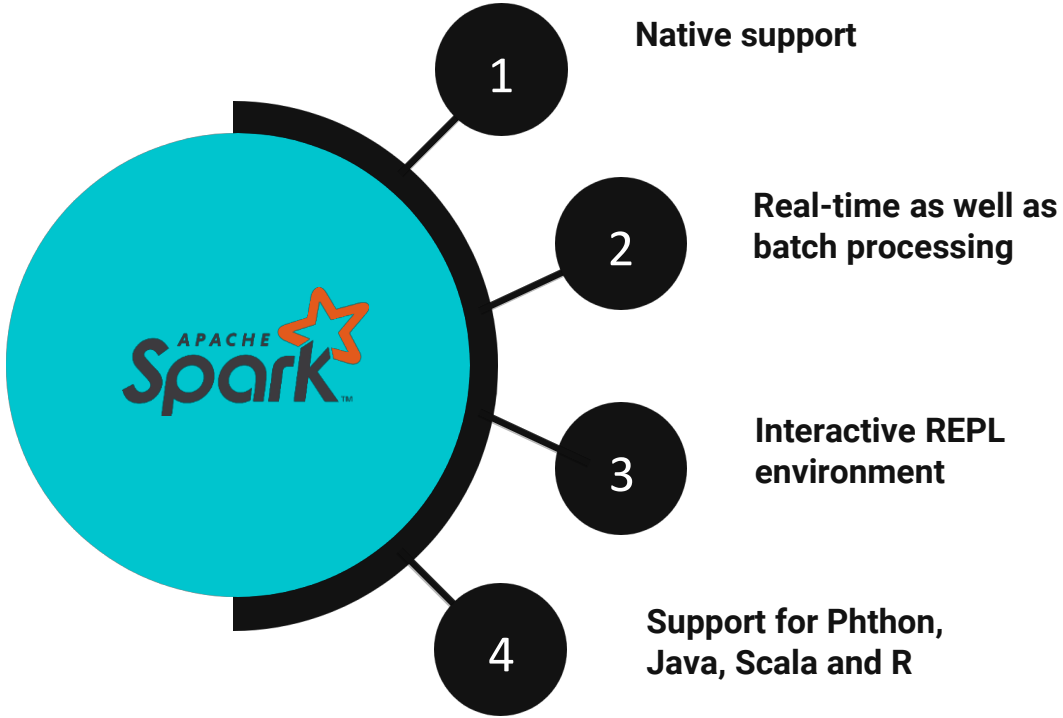
YARN takes care of resource allocation and figures out where and how to runt he job

3. Step Three

YARN stores the result in HDFS

Apache Spark Ecosystem




APACHE
Spark
1
Native support
2
Real-time as well as batch processing
3
Interactive REPL environment
4
Support for Phthon, Java, Scala and R

RDDs: Basic Building Blocks of Spark

"

RDD are still the fundamental building blocks of Spark

An RDD is a collection of entities – rows, records

RDDs: Basic Building Blocks of Spark

"

All operations in Spark are performed on in-memory objects

Two types of methods:

- Action - Return value
- Apply Transformation

Characteristics of RDDs

01

Partitioned

Split across data nodes in cluster

02

Immutable

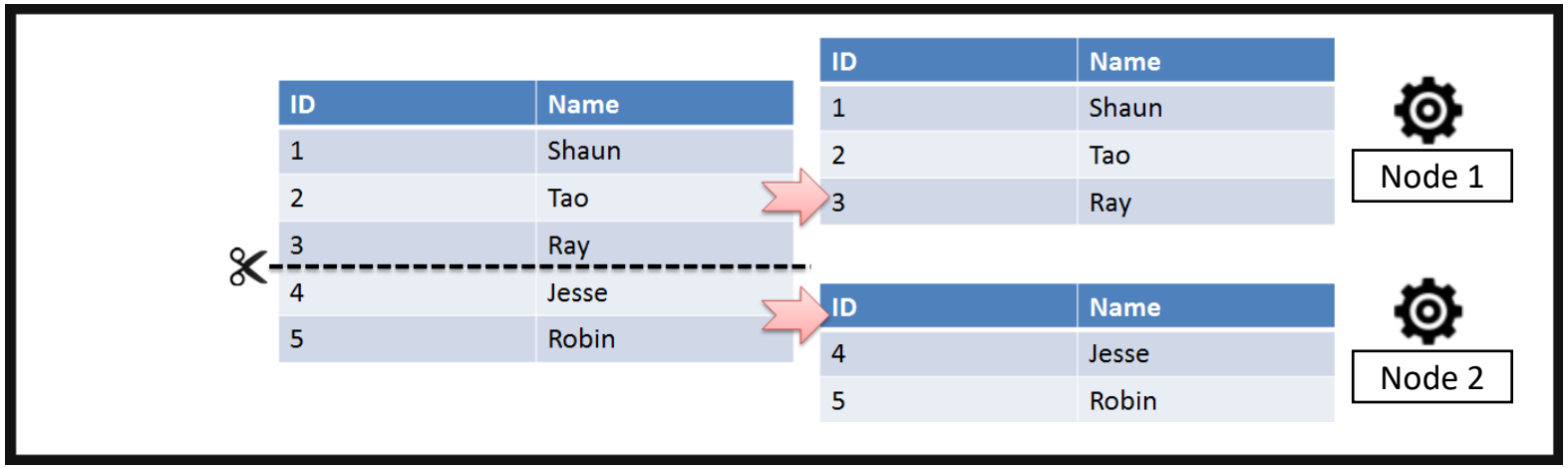
RDDs, once created, cannot be changed

03

Resilient

Can be reconstructed even if a node crashes

Partitioned



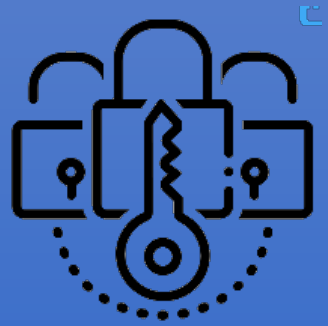
Parallel processing

The main reason for the splitting of data across multiple nodes is parallelization. Data can be processed in parallel on all of these individual nodes.

Data stored in memory for each node in cluster

The most important thing about Spark is that the contents of this RDD are kept entirely in memory across multiple cluster nodes.

- An RDD cannot be mutated
- Only **two types of operations** are permitted on RDD
  - **Transformation**: Transform in to another RDD
  - **Action**: Request a result



RDD: Immutable

RDD: Immutable

- A data set loaded in to RDD
- The user may define a chain of transformations on the dataset.

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

Example: Transformation

1. Load Data
2. Pick only the 3rd column
3. Sort the values

Transformation is executed only when a result is requested

Example: Action

1. The first 10 rows
2. A count
3. A Sum

Request a result using an action

RDD: Lazy Evaluation

- Spark keeps a record of the series of transformations requested by the user.
- It groups the transformations in an efficient way when an Action is requested

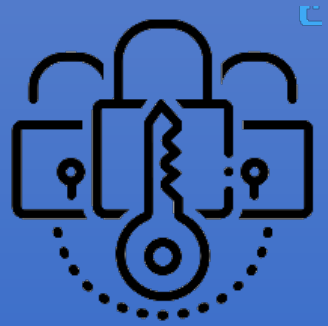
Lazy Evaluation

Spark keeps a record of the series of transformations requested by the user.

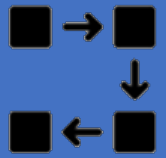
It groups the transformations in an efficient way when an Action is requested.

Transformation: Transform the RDD to create another RDD

Action: Read data from an RDD



RDD: Immutable



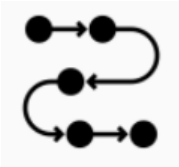
Resilient

Spark keeps a record of the series of transformations requested by the user

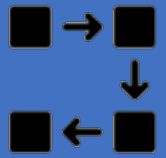
RDD can be created in 2 ways

- Reading a file
- Transforming another RDD

Every RDD keeps track of where it came from



This is RDD's lineage

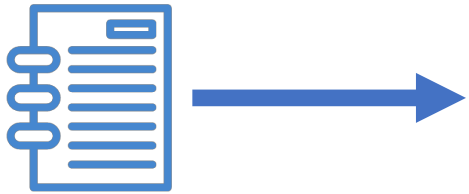


Resilient

Allows RDDs to be reconstructed when nodes crash

Allows RDDs to be Lazily instantiated (materialized) when accessing the results

DataFrame: Data in Rows and Columns

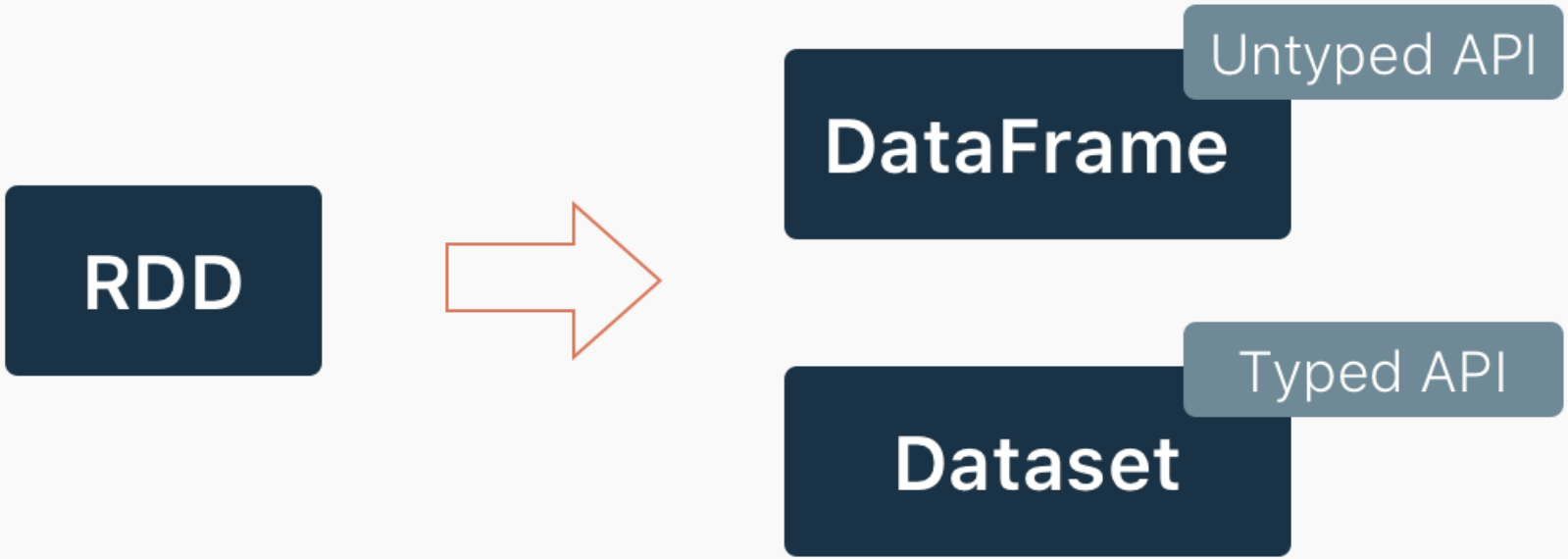


File

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

DataFrame

Apache Spark API



RDDs to Dataset

RDDs

Primary abstraction since initial versions

Immutable and distributed

Strong typing, use of Lambda

No optimized execution

Available in all languages

Datasets

Added to Spark in 1.6

Also immutable and distributed

Also support strong typing, lambdas

Leverage optimizers in recent versions

Present in Scala and Java, not python or R

Datasets to DataFrames

Datasets

Added to Spark in 1.6

Immutable and distributed

No named columns

Extension of DataFrames – OOP interface

Compile time type safety

Present in Scala, Java, not Python, R

DataFrames

Added to Spark in 1.3

Also immutable and distributed

Named columns, like Pandas or R

Conceptually equal to a table in an RDBMS

No type safety at compile time

Available in all languages

Starting Spark 2.0
APIs for Datasets and DataFrames heave merged

Datasets to DataFrames

Datasets

Scala and Java

Datasets of the Row() object in Scala/Java

often called DataFrames

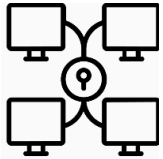
DataFrames

Python, R, Scala, Java

Equivalent to Dataset<Row> in Java or

Dataset[Row] in Scala

What makes Apache Spark difficult to use?

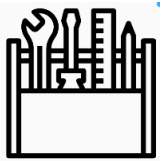


Infrastructure Management

Upgrade Challenge

User Interface

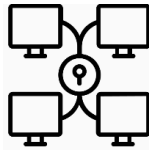
Manual Configuration



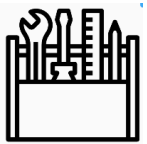
Tooling & Integration Complexity

Difficult to Collaborate on Projects

databricks



Efficient and Interactive Platform



Tools are available

Integrated and Interactive workspace

User Interface to manage Infrastructure (Scalability, failure recovery, upgrades)

Distributed processing of data

In-memory

Language support

- Scala, Python, SQL, R & Java

Use cases

- Batch & Stream processing
- Machine learning
- Advanced Analytics

An Apache Spark based Unified Analytics Platform, optimized for the cloud

1

Infrastructure Management

Setup and scale an optimized Apache Spark environment

2

Collaboration

Workspace for Data Engineers, Data Scientists & Business Analysts

3

Automation

Run data pipelines and workflows on demand or an schedule

4

Security

Built in access control and enterprise grade security

Microsoft Azure Databricks

A fast, easy, and collaborative Apache Spark™ based analytics platform optimized for Azure

Managed 1st Party Azure Service

Native integration with Azure & Its services;
Azure SLA and support

Transparency

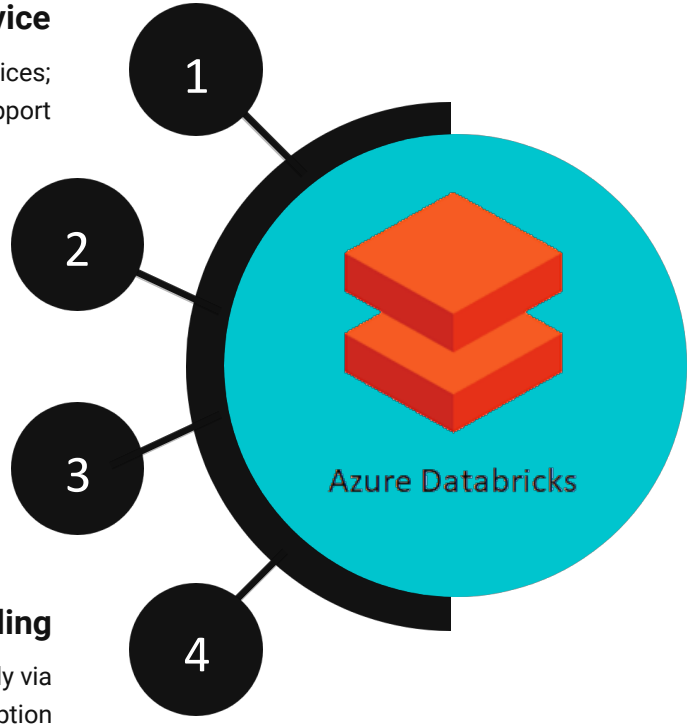
Deploys Databricks workspace and
clusters in customer subscription

Security

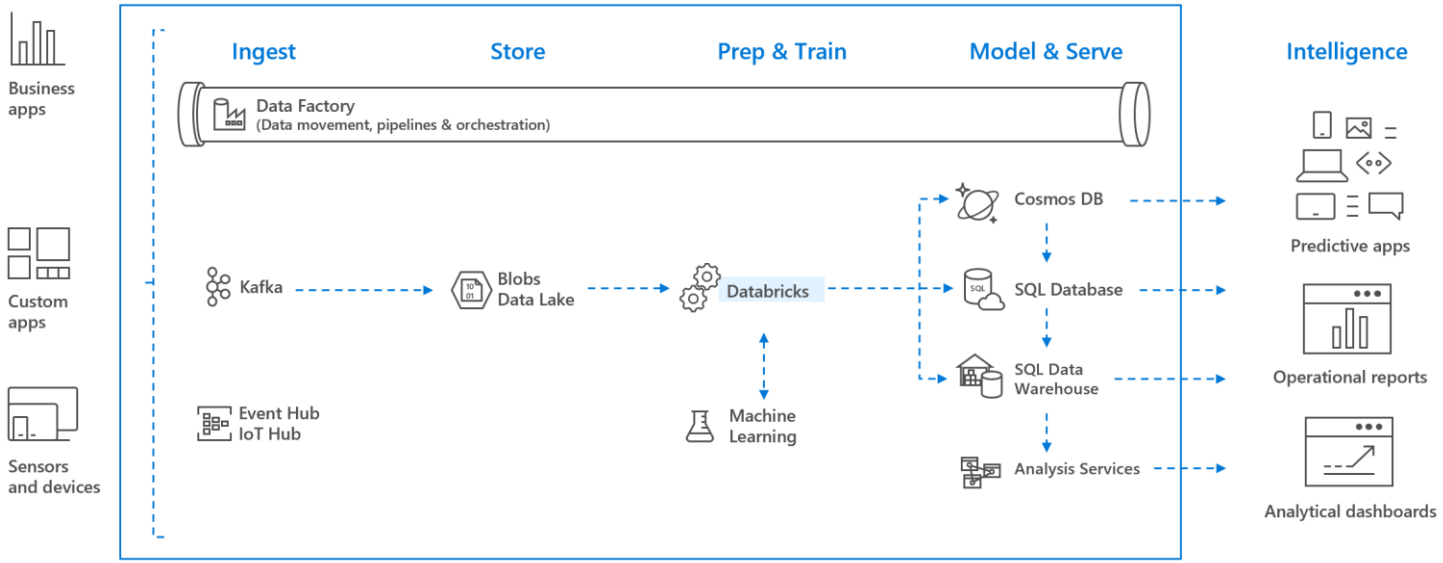
Natively integrates with Azure
Active Directory & Providers RBAC

United Billing

Pay for what you use only via
Azure subscription



Azure Databricks Architecture



Cluster Types

Interactive Cluster

Multiple users interactively analyze the data together

Job Cluster

Created and terminated for running automated jobs

Cluster Types

Interactive Cluster	Job Cluster
Interactively analyze the data	Run automated jobs
Created by users	Auto created when job starts
Manually terminate	Terminates when the job ends
Option to auto terminate, if inactive	Option to auto terminate not applicable
Low execution time	High throughput
Auto scale on demand	Auto scale on demand
Comparatively costly	Comparatively cheaper

Cluster Types

Standard Mode	High Concurrency Mode
Single user	Multiple users
No fault isolation	Fault isolation
No task preemption	Task preemption – fair resource sharing
Each user require separate cluster	Maximum cluster utilization
Supports Scala, Python, SQL, R % Java	Only supports Python, SQL & R

Cluster

There are two types of nodes

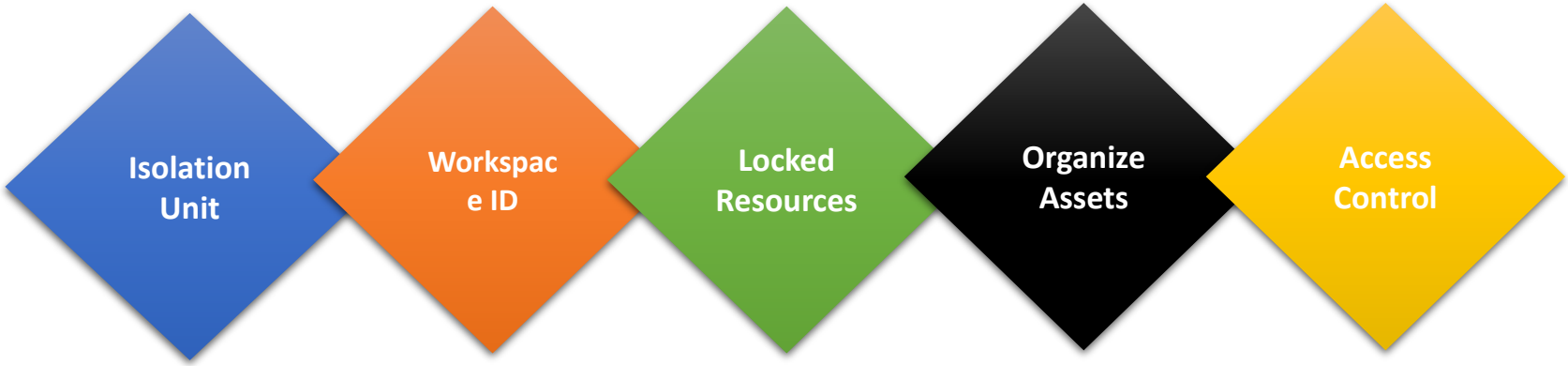
Worker Nodes

Multiple nodes perform data processing task

Driver Node

Distributes task to workers and coordinates execution

Workspace



Each workspace is isolated from others

Each workspace has an identifier

Deployed in control plane and data plane

Notebooks, Libraries, Dashboard etc.

Define access control on all assets

Notebooks

Code in any Spark supported Languages

Invoke notebook from others & pass data

Run directly on clusters or visa jobs

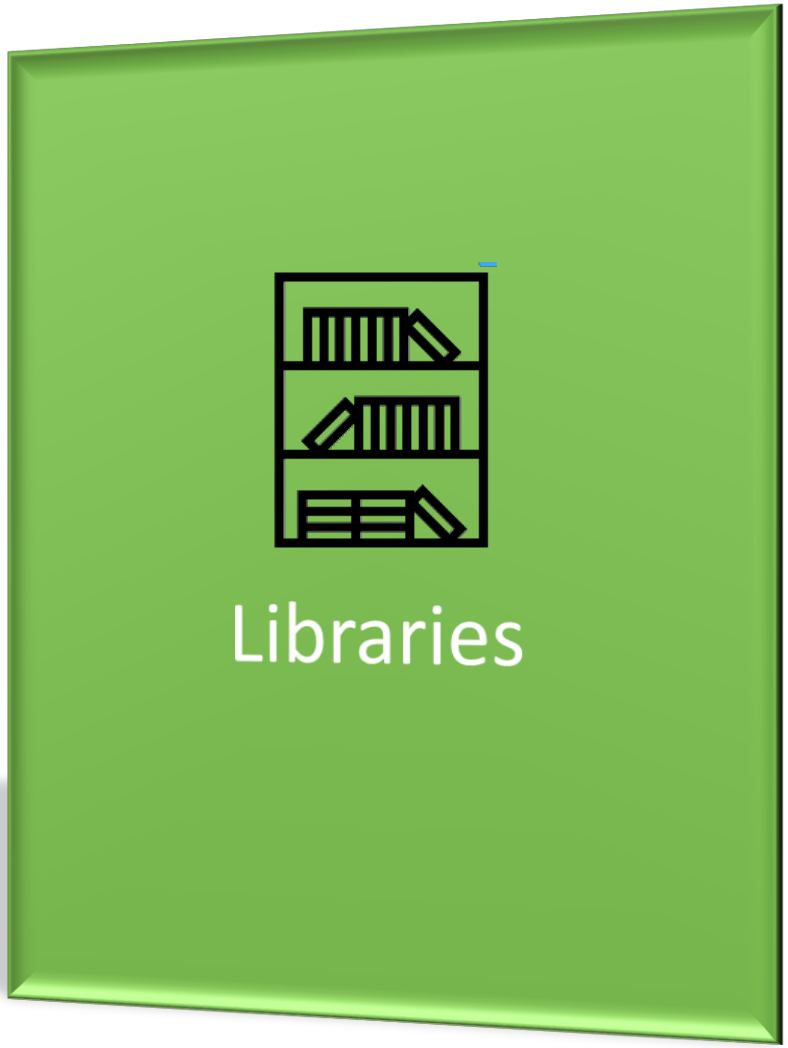
Turn data into graphs or build dashboards

Multiple users can edit and share comments

Jobs

- Execution of a notebook or JAR
- It can run immediately or on schedule
- Create job clusters to run jobs
- Each job can have different cluster configuration
- Monitor job runs and setup alerts

- Install 3rd party libraries
- Can be in any supported language
- Import the library into notebook to work
- Scoped at:
  - Cluster
  - Notebook



- Create databases and tables inside them
- Table:
  - Collection of structured data
  - Equivalent to DataFrame – perform same operations on table
  - Created using files lying on storage
  - Directly query or write to tables