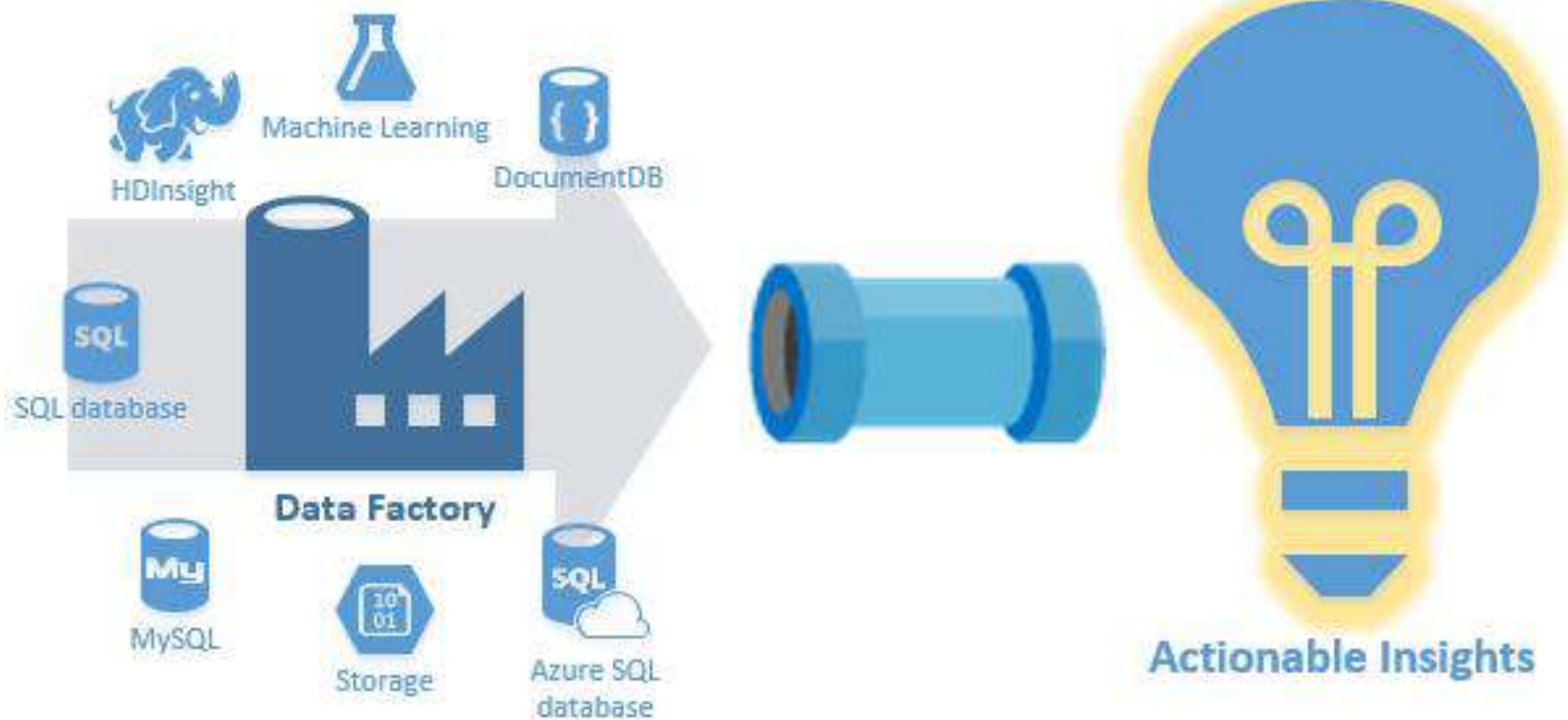


# Azure Data Factory



# Course Contents

- Introduction of Azure
- Introduction of Azure Data Factory
- Data Factory components
- Differences between v1 and v2
- Triggers
- Control Flow
- SSIS in ADFv2
- Demo

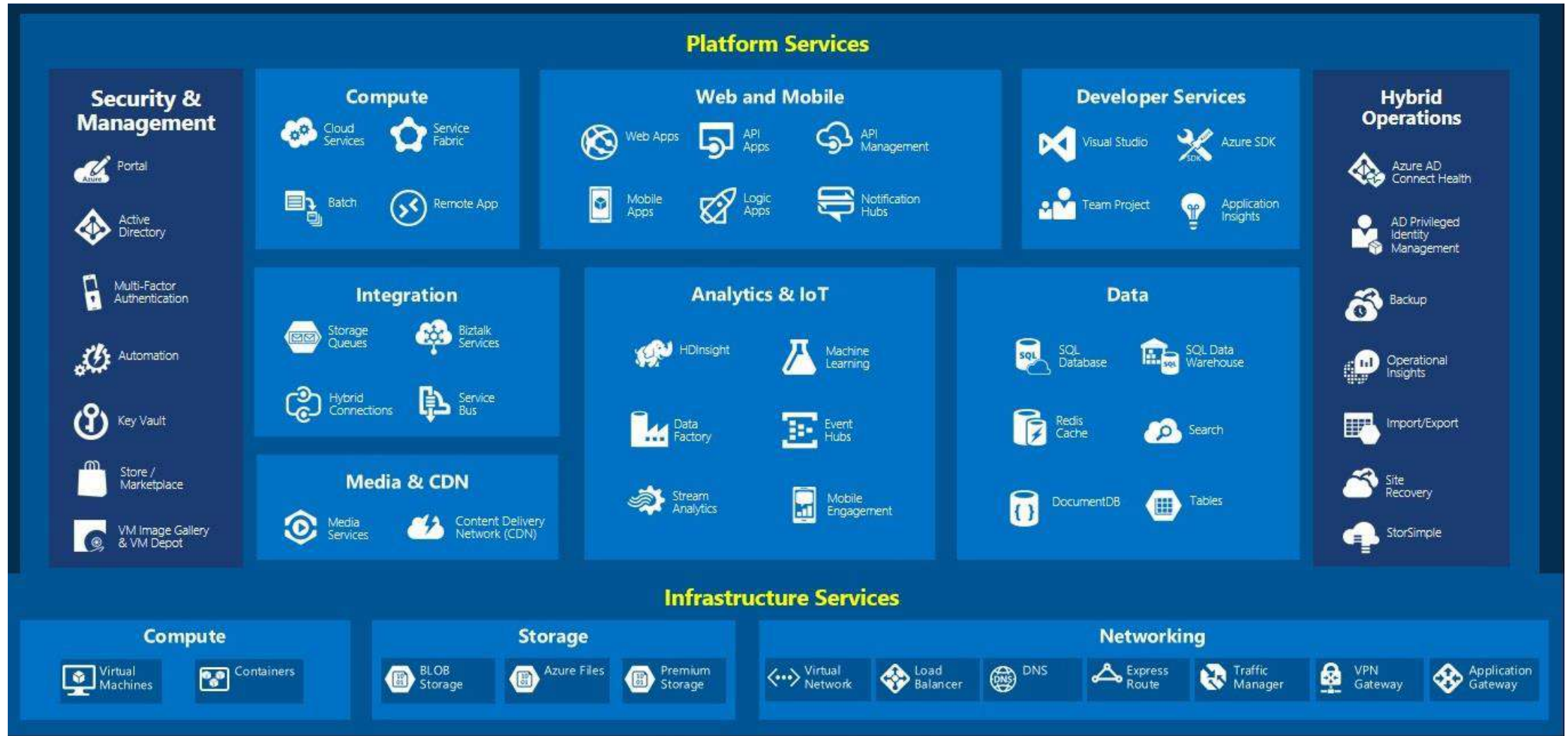
# Introduction of Azure

- Azure is Microsoft's cloud computing platform, provides cloud services that gives you the freedom to build, manage, and deploy applications on a massive global network using your favorite tools and frameworks.

[A quick explanation on how Azure works](#)

- Cloud computing is the delivery of computing services over the Internet using a **pay-as-you-go** pricing model. In other words it's a way to rent compute power and storage from someone's data center.
- Microsoft categorizes Azure cloud services into below product types:
  - Compute
  - Storage
  - Networking
  - Web
  - Databases
  - Analytics and IOT
  - Artificial Intelligence
  - DevOps

# Introduction of Azure



# Introduction of Azure Data Factory

- Azure Data Factory is a cloud-based data integration service to compose data storage, movement, and processing services into automated data pipelines.
- It compose of data processing, storage, and movement services to create and manage analytics pipelines, also provides orchestration, data movement and monitoring services.
- In the world of big data, raw, unorganized data is often stored in relational, non-relational, and other storage systems, big data requires service that can orchestrate and operationalize processes to refine these enormous stores of raw data into actionable business insights.
- Azure Data Factory is a managed cloud service that's built for these complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects.
- Azure Data Factory is a data ingestion and transformation service that allows you to load raw data from over 70 different on-premises or cloud sources. The ingested data can be cleaned, transformed, restructured, and loaded back into a data warehouse.
- Currently, there are two versions of the service: version 1 (V1) and version 2 (V2).

# Introduction of Azure Data Factory

- The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:



- Connect and collect:** The first step in building an information production system is to connect to all the required sources of data and processing, such as software-as-a-service (SaaS) services, databases, file shares, and FTP web services. The next step is to move the data as needed to a centralized location for subsequent processing.
- Transform and enrich:** After data is present in a centralized data store in the cloud, process or transform the collected data by using compute services such as HDInsight Hadoop, Spark, Data Lake Analytics, and Machine Learning.
- Publish:** After the raw data has been refined into a business-ready consumable form, load the data into Azure Data Warehouse, Azure SQL Database, Azure Cosmos DB, or whichever analytics engine your business users can point to from their business intelligence tools.
- Monitor:** After you have successfully built and deployed your data integration pipeline, providing business value from refined data, monitor the scheduled activities and pipelines for success and failure rates.

# Data Factory Components

- Azure Data Factory is composed of four key components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.
- **Pipeline:** A data factory might have one or more pipelines. A pipeline is a logical grouping of activities that performs a unit of work. For example, a pipeline can contain a group of activities that ingests data from an Azure blob, and then runs a Hive query on an HDInsight cluster to partition the data.
- **Activity:** Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.
- **Datasets:** Datasets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.
- **Linked services:** Linked services are much like connection strings, which define the connection information that's needed for Data Factory to connect to external resources. For example, an Azure Storage-linked service specifies a connection string to connect to the Azure Storage account.
- Linked services are used for two purposes in Data Factory :
  - To represent a data store that includes, but isn't limited to, an on-premises SQL Server database, Oracle database, file share, or Azure blob storage account.
  - To represent a compute resource that can host the execution of an activity. For example, the HDInsight Hive activity runs on an HDInsight Hadoop cluster.

# Data Factory Components

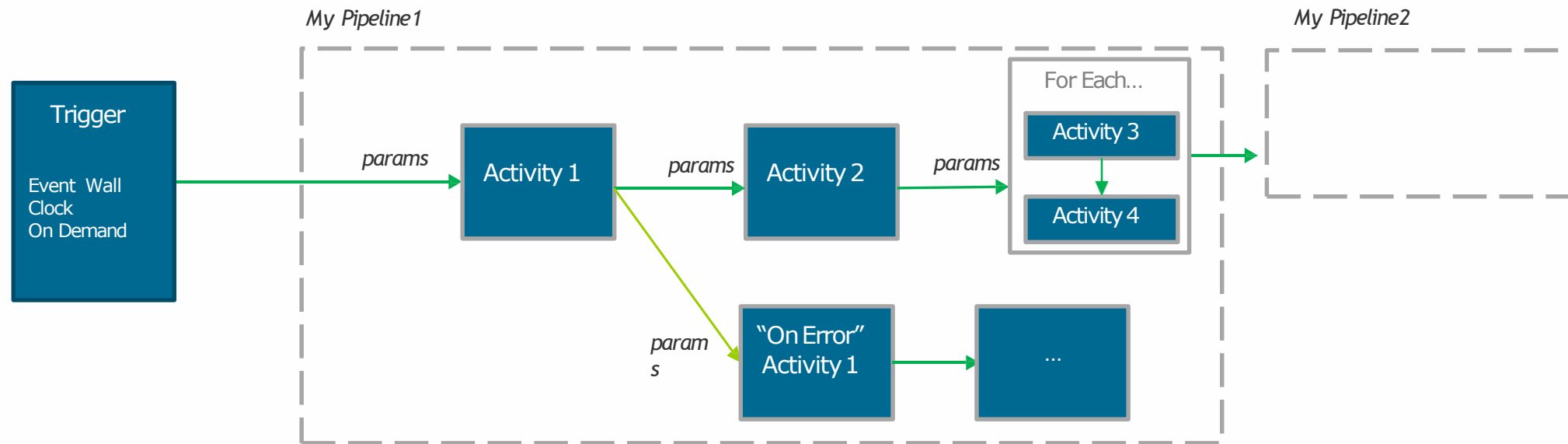
- Overview of Data Factory flow





# Data Factory Components

- Overview of Data Factory flow



# Data Factory Components

- Other components of Data Factory.
- **Triggers:** Triggers represent the unit of processing that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.
- **Pipeline runs:** A pipeline run is an instance of the pipeline execution. Pipeline runs are typically instantiated by passing the arguments to the parameters that are defined in pipelines. The arguments can be passed manually or within the trigger definition.
- **Parameters:** Parameters are key-value pairs of read-only configuration. Parameters are defined in the pipeline. Activities within the pipeline consume the parameter values.
- **Control flow:** Control flow is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, that is, For-each iterators.

# Differences between v1 and v2

Feature	Version 1	Version 2
Datasets	<p>A named view of data that references the data, can be utilized in activities as inputs and outputs.</p> <p>Datasets identify data within different data stores, such as tables, files, folders, and documents</p> <p><b>Availability</b> defines the processing window slicing model for the dataset (for example, hourly, daily, and so on).</p>	<p>Datasets are the same in the current version. However, you do not need to define <b>availability</b> schedules for datasets.</p>
Linked services	<p>Linked services are much like connection strings, which define the connection information that's necessary for Data Factory to connect to external resources.</p>	<p>Linked services are the same as in Data Factory V1, but with a new <b>connectVia</b> property to utilize the Integration Runtime compute environment of the current version of Data Factory.</p>

# Differences between v1 and v2

Feature	Version 1	Version 2
Pipelines	<p>A data factory can have one or more pipelines. A pipeline is a logical grouping of activities that together perform a task.</p>	<p>Pipelines are groups of activities that are performed on data. However, the scheduling of activities in the pipeline has been separated into new trigger resources.</p> <p>The Data Factory V1 concepts of <code>startTime</code>, <code>endTime</code>, and <code>isPaused</code> are no longer present in the current version of Data Factory.</p>
Activities	<p>Activities define actions to perform on your data within a pipeline. Data movement (copy activity) and data transformation activities (such as Hive, Pig, and MapReduce) are supported.</p>	<p>In this version of Data Factory, activities still are defined actions within a pipeline</p> <p>The current version of Data Factory introduces new control flow activities.</p>

# Differences between v1 and v2

Feature	Version 1	Version 2
Hybrid data movement and activity dispatch	Now called Integration Runtime, Data Management Gateway supported moving data between on-premises and cloud.	Data Management Gateway is now called Self-Hosted Integration Runtime. It provides the same capability as it did in V1. The Azure-SSIS Integration Runtime in the current version of Data Factory also supports deploying and running SQL Server Integration Services (SSIS) packages in the cloud.
Parameters	NA	Parameters are key-value pairs of read-only configuration settings that are defined in pipelines.

# Differences between v1 and v2

Feature	Version 1	Version 2
Expressions	Data Factory V1 allows to use functions and system variables in data selection queries and activity/dataset properties.	In this version of Data Factory, one can use expressions anywhere in a JSON string value.
Pipeline runs	NA	A single instance of a pipeline execution. Each pipeline run has a unique pipeline run ID. The pipeline run ID is a GUID that uniquely defines that particular pipeline run.
Activity runs	NA	An instance of an activity execution within a pipeline.
Trigger runs	NA	An instance of a trigger execution. For more information.
Scheduling	Scheduling is based on pipeline start/end times and dataset availability.	Scheduler trigger or execution via external scheduler.

# Differences between v1 and v2

Feature	Version 1	Version 2
Chaining activities	In V1, must configure the output of an activity as an input of another activity to chain them.	In this version of Data Factory, in the current version, one can chain activities in a sequence within a pipeline, by using the <b>dependsOn</b> property in an activity definition to chain it with an upstream activity.
Branching activities	NA	Can branch activities within a pipeline. The <b>If-condition</b> activity provides the same functionality that an if statement provides in programming languages.
Custom state passing	NA	Activity outputs including state can be consumed by a subsequent activity in the pipeline. By using this feature, we can build workflows where values can pass through activities.

# Differences between v1 and v2

Feature	Version 1	Version 2
Looping containers	NA	The ForEach activity defines a repeating control flow in your pipeline. This activity iterates over a collection and runs specified activities in a loop.
Trigger-based flows	NA	Pipelines can be triggered by on-demand (event-based, i.e. blob post) or wall-clock time.
Invoking a pipeline from another pipeline	NA	The Execute Pipeline activity allows a Data Factory pipeline to invoke another pipeline.
Delta flows	NA	A key use case in ETL patterns is “delta loads”. New capabilities in this current version, such as lookup activity, flexible scheduling, and control flow, enable this use case.



# Differences between v1 and v2

Feature	Version 1	Version 2
Other control flow activities	NA	ForEach activity, Web activity, Lookup activity, Get metadata activity, Wait activity.
Deploy SSIS packages to Azure	NA	We can Azure-SSIS if you want to move our SSIS workloads to the cloud, create a data factory by using the current version, and provision an Azure-SSIS Integration Runtime.
Custom activities	In V1, we implement (custom) DotNet activity code by creating a .NET class library project with a class that implements the Execute method of the IDotNetActivity interface. Therefore, you need to write your custom code in .NET Framework 4.5.2 and run it on Windows-based Azure Batch Pool nodes.	In a custom activity in this version, you don't have to implement a .NET interface. You can directly run commands, scripts, and your own custom code compiled as an executable.

# Triggers

How do pipelines get started

1. on-demand
2. Wall-clock Schedule
3. Tumbling Window (aka time-slices in v1)
4. *Event*

# Triggers

## How do pipelines get started

### 1. Power Shell:

Invoke-AzureRmDataFactoryV2Pipeline +Parameters

2. Rest API: <https://management.azure.com/subscriptions/mySubId/resourceGroups/myResourceGroup/providers/Microsoft.DataFactory/factories/{yourDataFactory}/pipelines/{yourPipeline}/createRun?api-version=2017-03-01-preview>

### 3. NET:

client.Pipelines.CreateRunWithHttpMessagesAsync(+ parameters)

### 4. Azure Portal

(Data factory -> <Author & Monitor> -> Pipeline runs)

# Triggers

## Run pipeline by schedule

```
{
  "properties": {
    "type": "ScheduleTrigger",
    "typeProperties": {
      "recurrence": {
        "frequency": <<Minute, Hour, Day, Week, Year>>,
        "interval": <<int>>, // optional, how often to fire (default to 1)
        "startTime": <<datetime>>,
        "endTime": <<datetime>>,
        "timeZone": "UTC"
      }
      "schedule": { // optional (advanced scheduling specifics)
        "hours": [<<0-24>>],
        "weekDays": ": [<<Monday-Sunday>>],
        "minutes": [<<0-60>>],
        "monthDays": [<<1-31>>],
        "monthlyOccurrences": [
          {
            "day": <<Monday-Sunday>>,
            "occurrence": <<1-5>>
          }
        ]
      }
    }
  }
}
```

Description

Type \*

ScheduleTrigger

Start Date (UTC) \*

11/23/2018 9:39 AM

Recurrence \*

Every 1 Day(s)

Advanced recurrence options

Execute at these times

Hours (UTC) 6 \* 9 \* 12 \*

Minutes (UTC) 30 \*

Schedule execution times

06:30,09:30,12:30

End \*

☒ No End ☐ On Date

Annotations

+ New

☒ Activated

Cancel

Finish

# Triggers

## Tumbling Window

Tumbling window triggers are a type of trigger that fires at a periodic time interval from a specified start time, while retaining state. Tumbling windows are a series of fixed-sized, non-overlapping, and contiguous time intervals.

```
{
  "name": "MyTriggerName",
  "properties": {
    "type": "TumblingWindowTrigger",
    "runtimeState": "<<Started/Stopped/Disabled - readonly>>",
    "typeProperties": {
      "frequency": "<<Minute/Hour>>",
      "interval": <<int>>,
      "startTime": "<<datetime>>",
      "endTime": "<<datetime - optional>>\"",
      "delay": "<<timespan - optional>>",
      "maxConcurrency": <<int>> (required, max allowed: 50),
      "retryPolicy": {
        "count": <<int - optional, default: 0>>,
        "intervalInSeconds": <<int>>,
      }
    }
  },
  "pipeline": {
    "pipelineReference": {
      "type": "PipelineReference",
      "referenceName": "MyPipelineName"
    }
  },
}
```

← New Trigger

Name \*

TumblingWindowTrigger

Description

Type \*

☐ Schedule ☒ Tumbling Window ☐ Event

Start Date (UTC) \*

07/27/2018 5:49 AM

Recurrence \*

Every Minute Every 15 Minute(s)

End \*

☒ No End ☐ On Date

Advanced

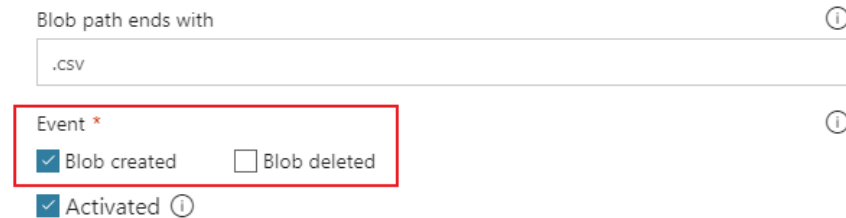
☒ Activated

Cancel Next

# Triggers

## Event Based Trigger

Data integration scenarios often require Data Factory customers to trigger pipelines based on events. Data Factory is now integrated with Azure Event Grid, which lets you trigger pipelines on an event.



Blob path ends with ⓘ

.CSV

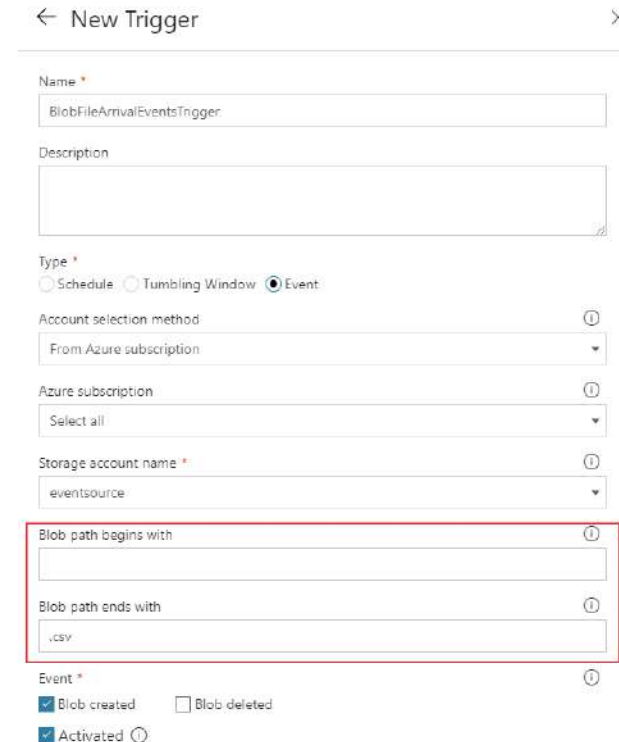
Event \* ⓘ

☒ Blob created ☐ Blob deleted

☒ Activated ⓘ

Cancel

Next



← New Trigger X

Name \*

BlobFileArrivalEventsTrigger

Description

Type \*

☐ Schedule ☐ Tumbling Window ☒ Event ⓘ

Account selection method ⓘ

From Azure subscription ▼

Azure subscription ⓘ

Select all ▼

Storage account name \* ⓘ

eventsource ▼

Blob path begins with ⓘ

Blob path ends with ⓘ

.CSV

Event \* ⓘ

☒ Blob created ☐ Blob deleted

☒ Activated ⓘ

Cancel

Next

# Control Flow

## Activities Known from v1 - Data Transformation Activities

Data transformation activity	Compute environment
Hive	HDInsight [Hadoop]
Pig	HDInsight [Hadoop]
MapReduce	HDInsight [Hadoop]
Hadoop Streaming	HDInsight [Hadoop]
Spark	HDInsight [Hadoop]
Machine Learning activities: Batch Execution and Update Resource	Azure VM
Stored Procedure	Azure SQL, Azure SQL Data Warehouse, or SQL Server
U-SQL	Azure Data Lake Analytics

# Control Flow

## New! Control Flow Activities in v2

Control activity	Description
Execute Pipeline Activity	allows a Data Factory pipeline to invoke another pipeline.
ForEachActivity	used to iterate over a collection and executes specified activities in a loop.
WebActivity	call a custom REST endpoint and pass datasets and linked services
Lookup Activity	look up a record/ table name/ value from any external source to be referenced by succeeding activities. Could be used for incremental loads!
Get Metadata Activity	retrieve metadata of any data in Azure Data Factory e.g. did another pipeline finish
Do Until Activity	similar to Do-Until looping structure in programming languages.
If Condition Activity	do something based on condition that evaluates to true or false.

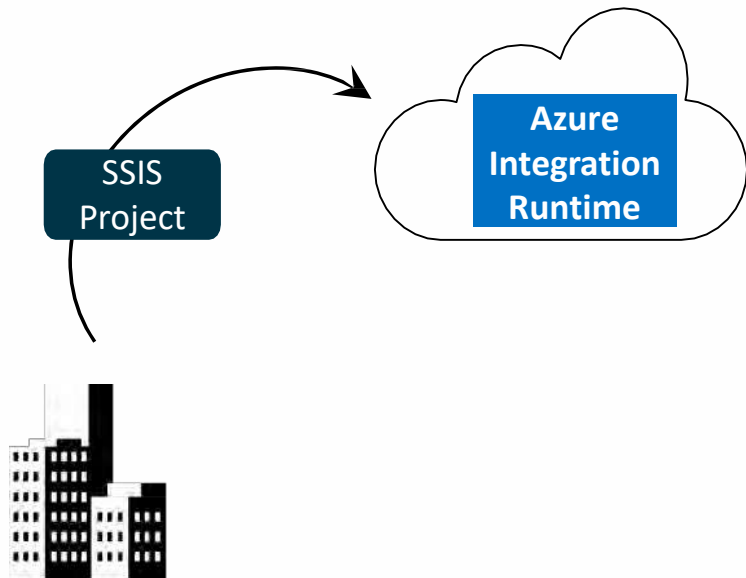


# Control Flow

## New! Control Flow Activities in v2

Control activity	Description
Append Variable Activity	to add a value to an existing array variable defined in a Data Factory pipeline.
Filter activity	to apply a filter expression to an input array.
Set Variable Activity	to set the value of an existing variable of type String, Bool, or Array defined in a Data Factory pipeline.
Validation activity	to ensure the pipeline only continues execution once it has validated the attached dataset reference exists
Wait activity	the pipeline waits for the specified period of time before continuing with execution of subsequent activities.
Webhook activity	to control the execution of pipelines through your custom code.
Data flow activity	to run your ADF data flow in pipeline debug (sandbox) runs and in pipeline triggered runs. (This Activity is in public preview)

# SSIS in ADFv2



## Managed Cloud Environment

Pick # nodes & node size

Resizable

SQL Standard Edition, Enterprise coming soon

## Compatible

Same SSIS runtime across Windows, Linux, Azure Cloud

## SSIS + SQL Server

SQL Managed instance + SSIS (in ADFv2) Access on premises data via VNet

## Get Started

Hourly pricing (no SQL Server license)

# SSIS in ADFv2

## Integration runtime - Different capabilities

### **1. Data Movement**

Move data between data stores, built-in connectors, format conversion, column mapping, and performant and scalable data transfer

### **2. Activity Dispatch**

Dispatch and monitor transformation activities (e.g. Stored Proc on SQL Server, Hive on HD Insight..)

### **3. SSIS package execution**

Execute SSIS packages

# SSIS in ADFv2

## Combinations of IR types, networks and capabilities

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

# SSIS in ADFv2

## Integration runtimes

### 1. Azure Integration Runtime

- move data between cloud data stores
- fully managed
- serverless compute service (PaaS) on Azure
- cost will occur only for time of duration
- user could define data movement units
- compute size auto scaled for copy jobs

# SSIS in ADFv2

## Integration runtimes

### 2. Self-hosted Integration Runtime

- perform data integration securely in a private network environment w/o direct line-of-sight from the public cloud environment
- Installed on-premises in your environment
- Supports copy activity between a cloud data stores and a data store in private network
- Supports dispatching the transform activities
- Works in your corporate network or virtual private network<sup>30</sup>
- Only Outbound http based connections to open internet
- Scale out supported

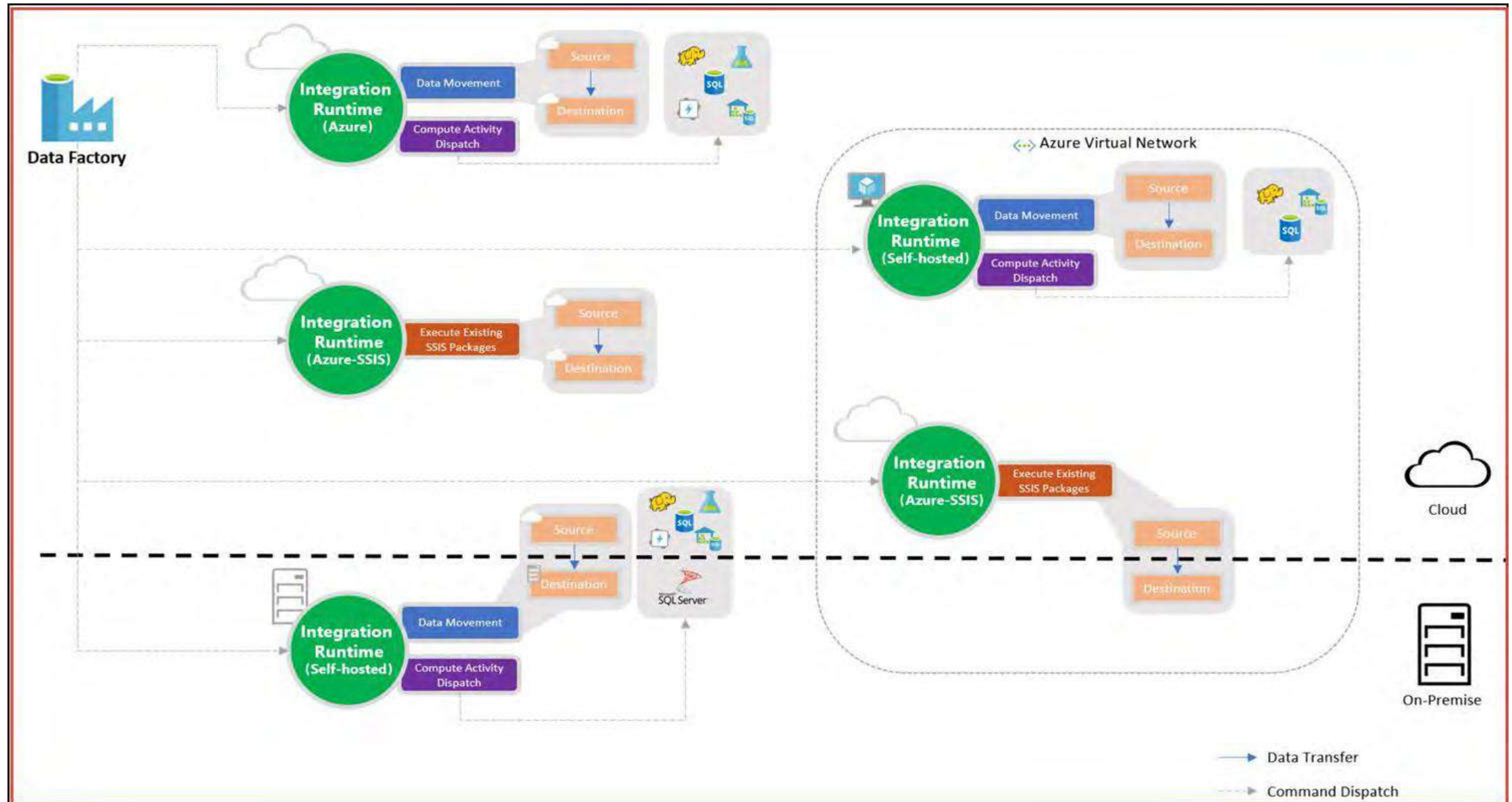
# SSIS in ADFv2

## Integration runtimes

### 3. Azure-SSIS Integration Runtime

- fully managed cluster of Azure VMs for native execution of SSIS packages.
- Access to on-premises data access using Vnet (classic in preview)
- SSIS Catalog on Azure SQL DB or SQL Managed Instance
- scale up: set node size
- scale out: number of nodes
- reduce costs by start/stop of service

# SSIS in ADFv2





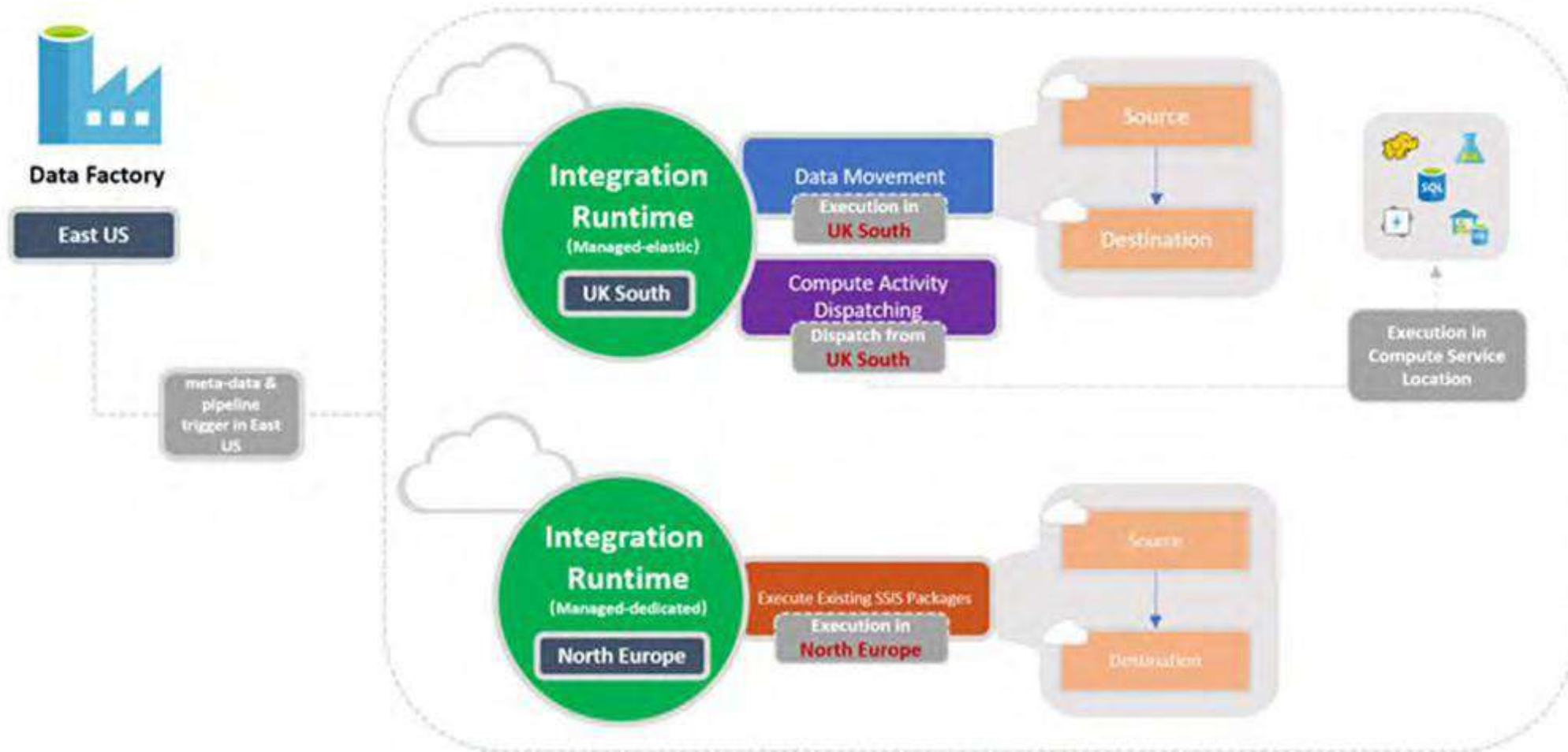
# SSIS in ADFv2

## Determining which IR to use

- IR is referenced as linked service in the data factory
- Transformation Activity: target compute needs linked service
- Copy Activity: source and sink need linked service, the computation is determined automatically (see detail on msdn)
- Integration runtime locations can differ from its Data Factory location which uses it

# SSIS in ADFv2

## Samples ADF and IR locations



# SSIS in ADFv2

## Scaleable **Integration** Services

How to scale up/out using 3 Settings on Azure SSIS IR

1. Configurable number of nodes on which SSIS is executed

`$AzureSSISNodeSize = "Standard_A4_v2"` # minimum size, others avail.\*

2. Configurable size of nodes

`$AzureSSISNodeNumber = 2` #between 1 and 10 nodes\*

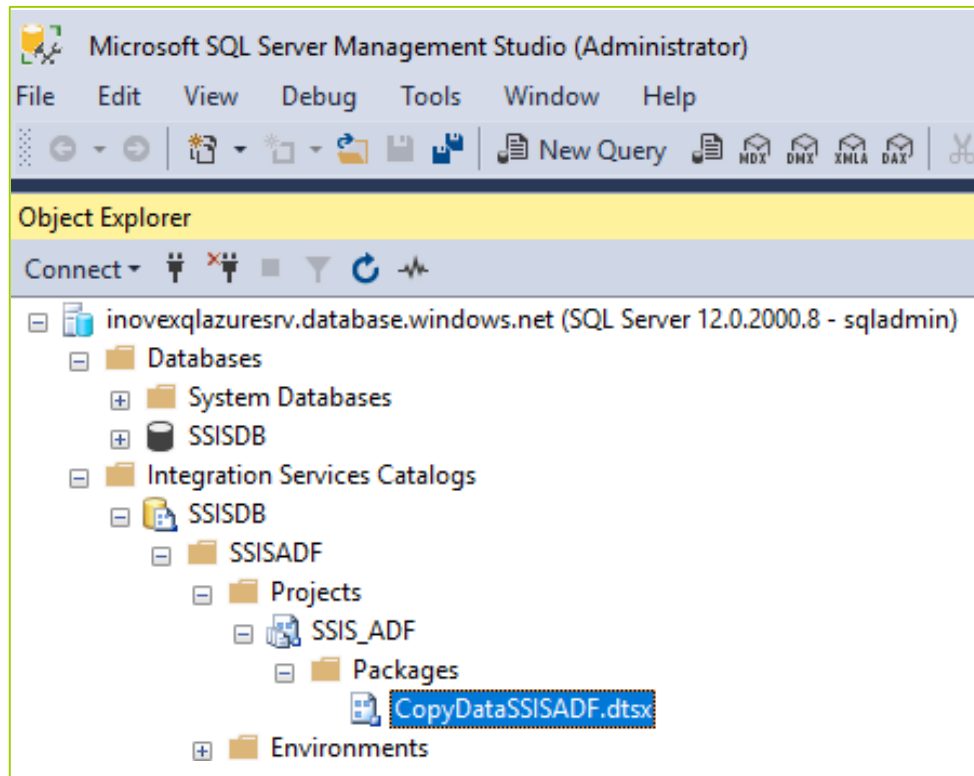
3. Configurable maximum parallel executions pernode

`$AzureSSISMaxParallelExecutionsPerNode = 2` # between 1-8\*

In Powershell CommandLet: Set-

`AzureRmDataFactoryV2IntegrationRuntime`

# SSIS in ADFv2



Execution Overview

Filter: Result: All; (3 more)

Result	Duration (sec)	Package Name	Task Name	Execution Path
Succeeded	17.297	CopyDataSSISADF.dtsx	CopyDataSSISADF	<a href="#">\CopyDataSSISADF</a>
Succeeded	17.234	CopyDataSSISADF.dtsx	Data Flow Task	<a href="#">\CopyDataSSISADF\Data Flow Task</a>

# SSIS in ADFv2

## Notes from the field

1. Connect in SSMS directly to the DB SSISDB to see SSIS Catalog
2. Deploy from Visual Studio only in Project Deployment Mode, workaround SSMS Import
3. In Preview no SSIS 3rd Party components supported (e.g. Theobald for SAP, cozyroc..)
4. V2 IR supports only V2 pipelines and V1 IR supports only V1. Both cannot be used interchangeably. Even though it is the same installer
5. Use same location for Azure-SSIS IR and the used (SQL<sup>37</sup> Azure) DB for SSIS Catalog

# SSIS in ADFv2

## Execution Methods

1. SSIS packages can be executed via SSMS
2. SSIS packages can be executed via CLI
  - › Run dtexec.exe from the command prompt
3. SSIS packages can be executed via custom code/PSH using SSIS MOM .NET SDK/API
  - › Microsoft.SqlServer.Management.IntegrationServices.dll is installed in .NET GAC with SQL Server/SSMS installation
4. SSIS packages can be executed via T-SQL scripts executing SSISDB sprocs
  - › Execute SSISDB sprocs [catalog].[create\_execution] + [catalog].[set\_execution\_parameter\_value] + [catalog].[start\_execution]

# SSIS in ADFv2

## Scheduling Methods

1. SSIS package executions can be directly/explicitly scheduled via ADFv2 App (Work in Progress)
  - › For now, SSIS package executions can be indirectly/implicitly scheduled via ADFv1/v2 Sproc Activity
2. If you use Azure SQL MI server to host SSISDB
  - › SSIS package executions can also be scheduled via Azure SQL MI Agent (Extended Private Preview)
3. If you use Azure SQL DB server to host SSISDB
  - › SSIS package executions can also be scheduled via Elastic Jobs (Private Preview)
4. If you keep on-prem SQL Server
  - › SSIS package executions can also be scheduled via on-prem SQL Server Agent

# Demo

