IntelliPaat

Designing and Implementing an Azure Data Solution

DP 200 and DP 201

IntelliPaat

Azure Batch and Azure Data Factory

Agenda

01 What is Azure Batch?

02 What is Azure Data Factory?

03 Why Azure Data Factory?

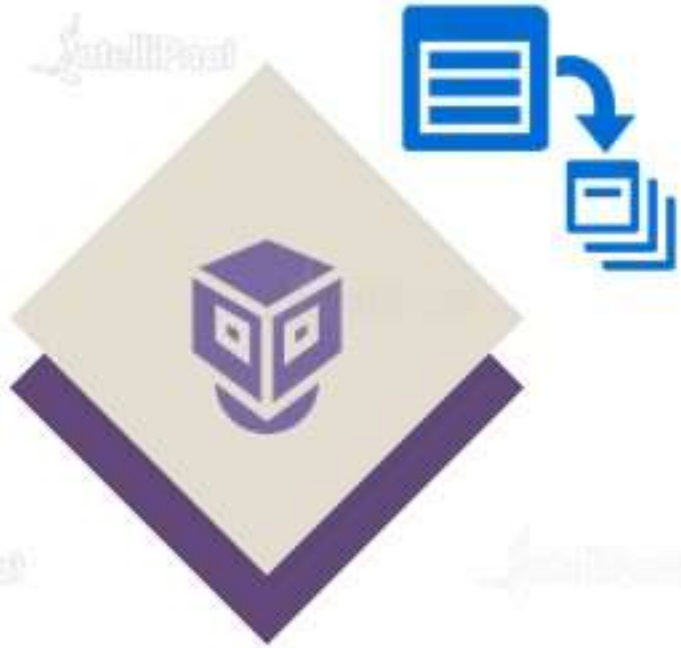
04 Integration Runtime in Azure Data Factory

05 Mapping Data Flows

What is Azure Batch?

What is Azure Batch?

We use Azure Batch to run large-scale parallel and high-performance computing batch jobs efficiently in Azure



Azure Batch creates and manages a pool of compute nodes (virtual machines), installs applications we want to run, and schedules jobs to run on the nodes

We use Batch APIs and tools, command-line scripts, or Azure Portal to configure, manage, and monitor jobs

What is Azure Batch?

For example, we can build a service with Batch to run a Monte Carlo risk simulation for a financial services company or a service that processes images

Developers can use Batch as a platform service to build SaaS applications or client apps where large-scale execution is required

No additional charges for using Batch. We have to pay only for the underlying resources consumed, such as virtual machines, storage, and networking

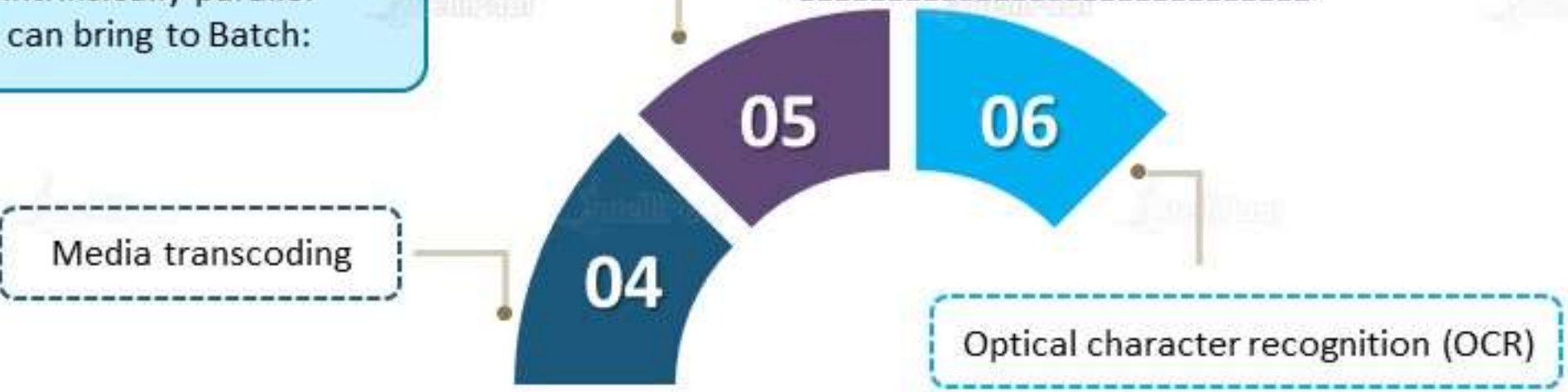
Intrinsically Parallel Workloads

When applications are executing, they might access some common data, but they do not communicate with other instances of the applications

Intrinsically parallel workloads are those wherein applications can run independently, and each instance completes a part of the work

Intrinsically Parallel Workloads

Some examples of intrinsically parallel workloads that we can bring to Batch:



Genetic sequence analysis

Media transcoding

Optical character recognition (OCR)

Image analysis and processing

Financial risk modeling using Monte Carlo simulations

VFX and 3D image rendering

Tightly Coupled Workloads

01

Tightly coupled workloads are those wherein the applications we run need to communicate with each other

02

Tightly coupled applications normally use the Message Passing Interface API

03

We can run our tightly coupled workloads with Batch using Microsoft MPI or Intel MPI

Tightly Coupled Workloads

Some examples of tightly coupled workloads:

Finite element analysis

Fluid dynamics

Multi-node AI training

Tightly Coupled Workloads

- Many tightly coupled jobs can be run in parallel using Batch
- For example, we perform multiple simulations of a liquid flowing through a pipe with varying pipe widths

Additional Batch Capabilities

High-level, workload-specific capabilities are also available in Azure Batch

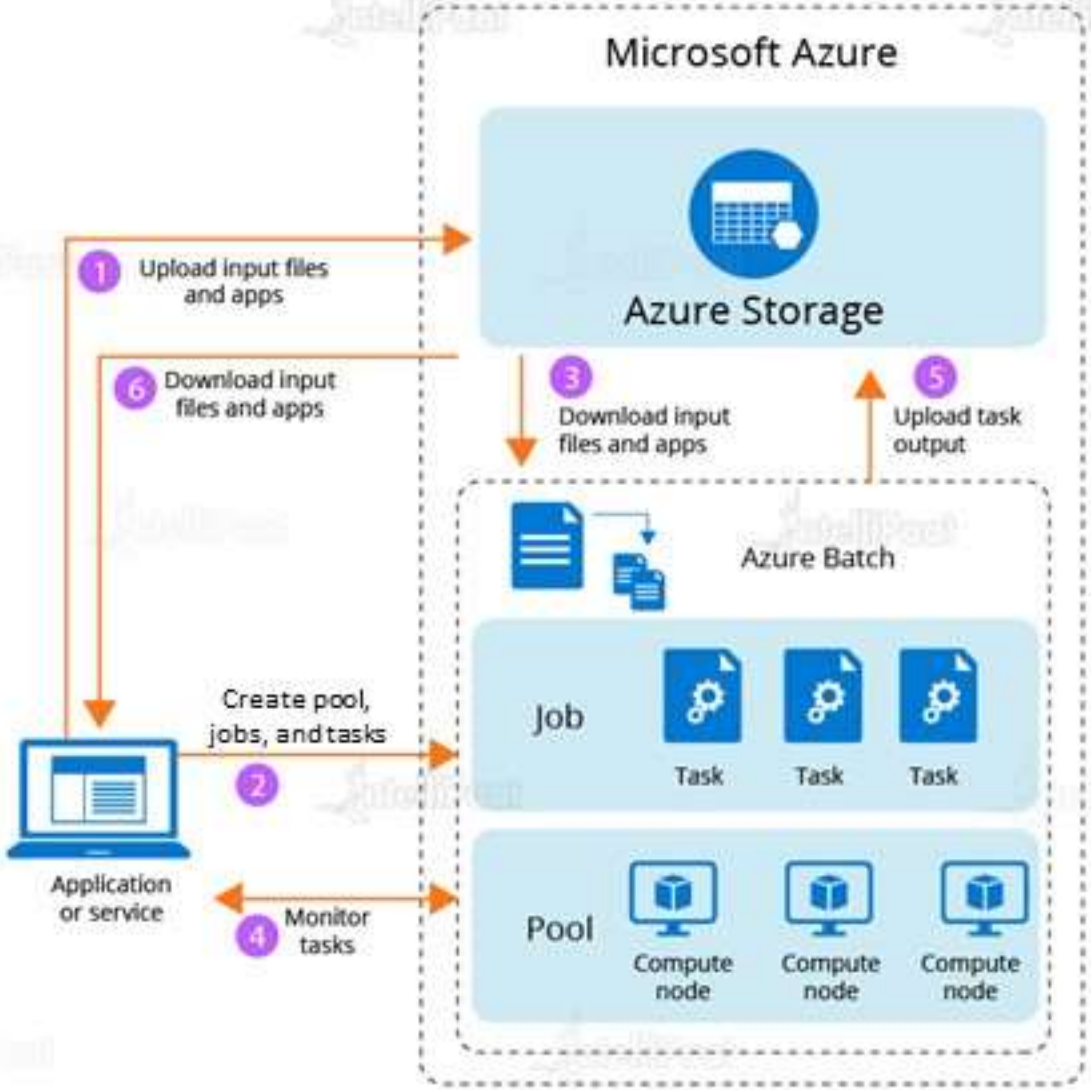
Batch supports large-scale rendering workloads with rendering tools including Autodesk Maya, 3ds Max, Arnold, and V-Ray

R users can install the **doAzureParallel** package to easily scale out the execution of R algorithms on Batch pools

Additional Batch Capabilities

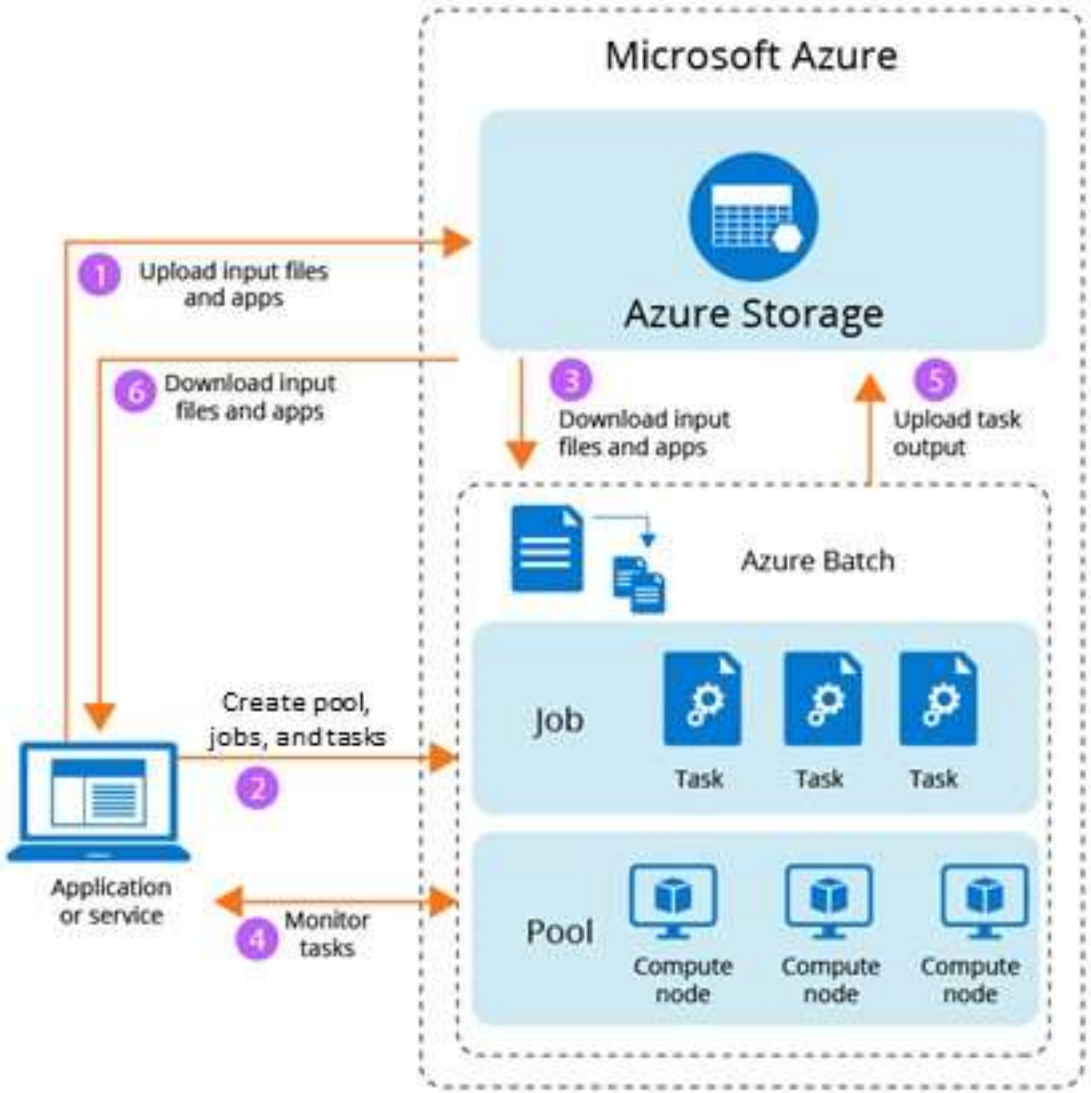
We can also run Batch jobs as part of a larger Azure workflow to transform data, managed by tools such as **Azure Data Factory**

Working of Azure Batch



- A common scenario for Batch involves scaling up intrinsically parallel work, such as the rendering of images for 3D scenes on a pool of compute nodes
- The diagram shows the steps in a common Batch workflow, with a client application or hosted service using Batch to run the parallel workload

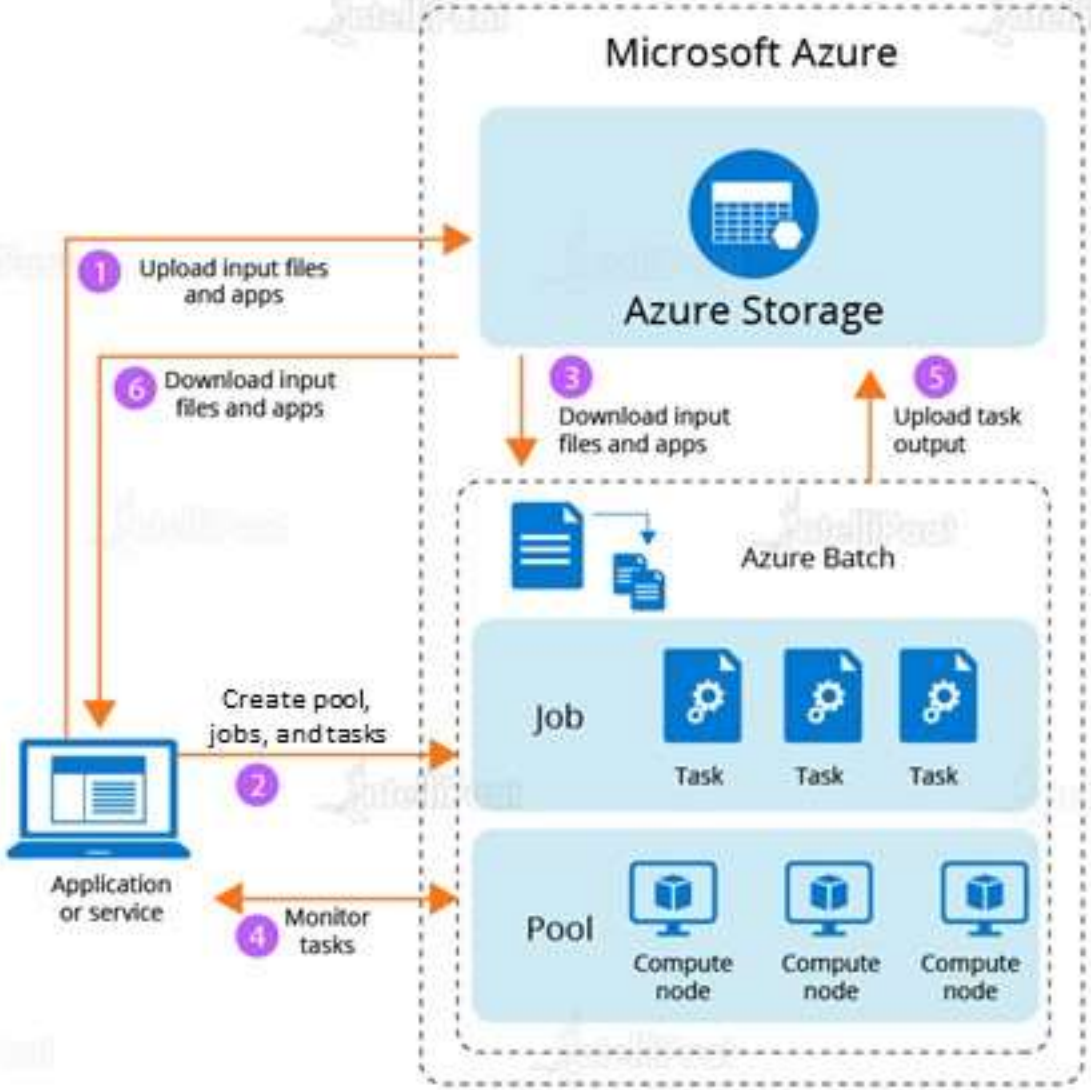
Working of Azure Batch



Step 1: Upload input files and applications to the our Azure Storage account

- **Input:** Any data that our application processes, such as financial modeling data or video files to be transcoded
- **Application files:** Scripts or applications that process data, such as a media transcoder

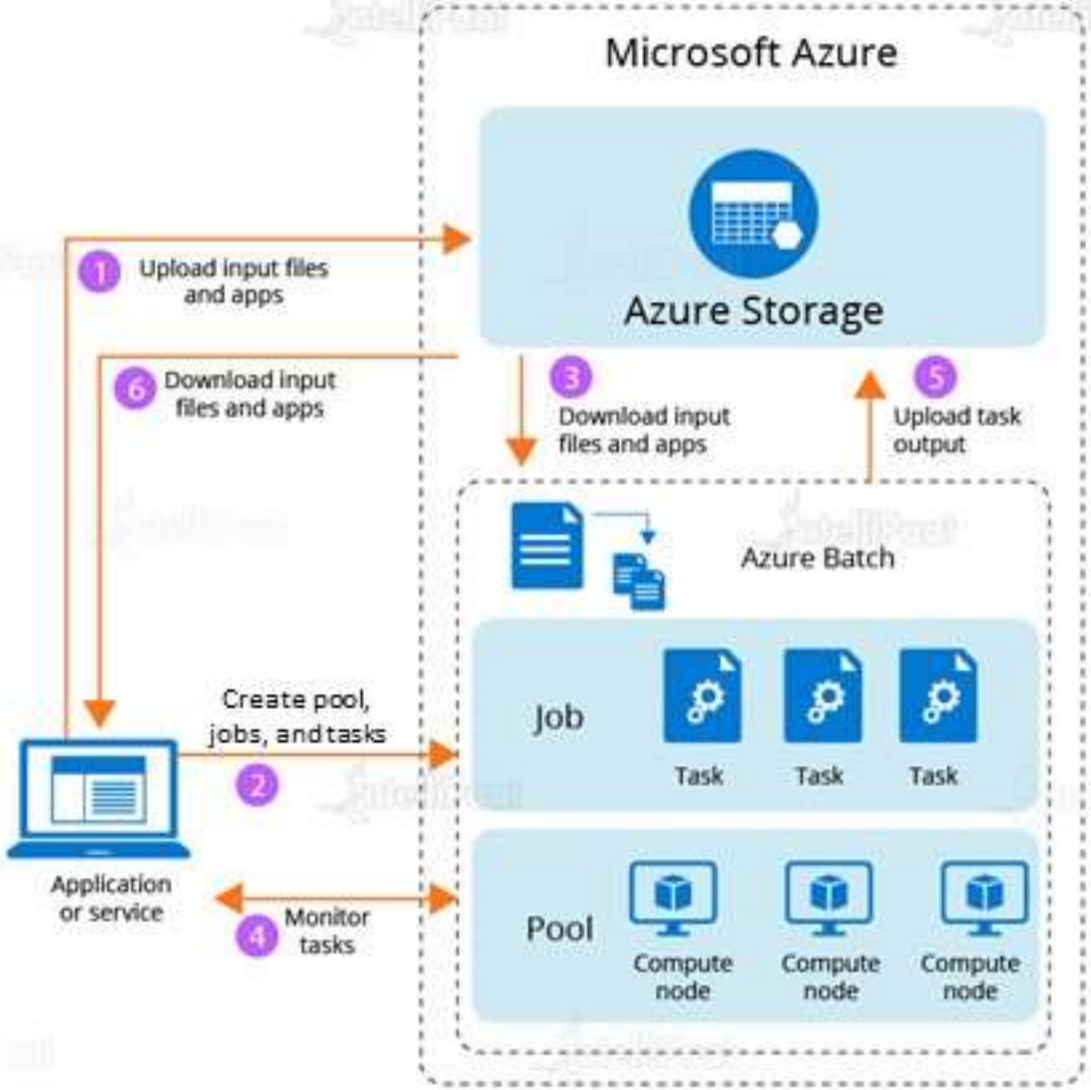
Working of Azure Batch



Step 2: Create a **Batch pool** of compute nodes in our Batch account, a **job** to run the workload on the pool, and **tasks** in the job

- Pool nodes are the VMs that execute our tasks
- When we add tasks to a job, the Batch service automatically schedules the tasks for execution on the compute nodes in the pool

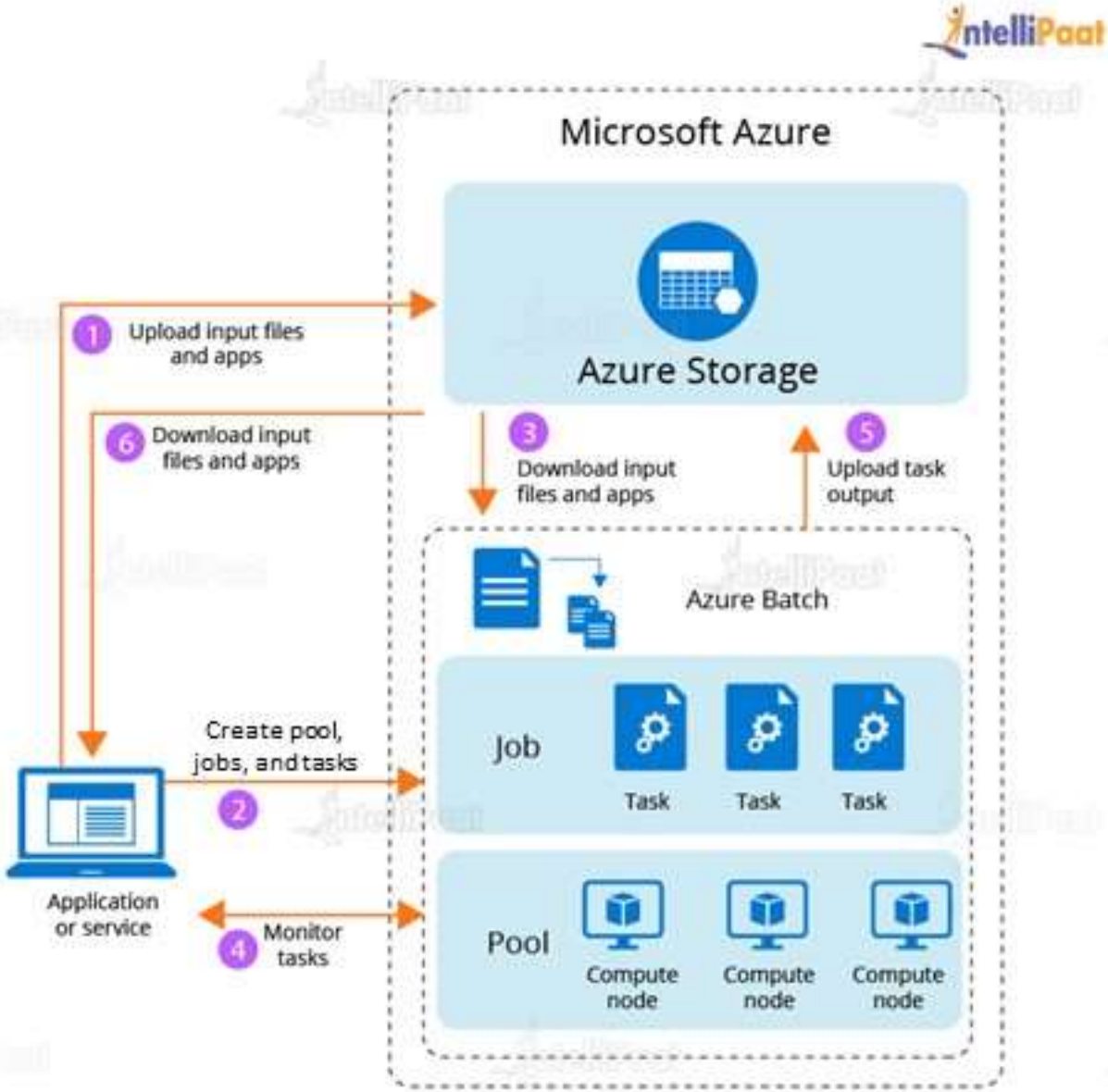
Working of Azure Batch



Step 3: Download input files and applications to Batch

- When downloading from Azure Storage gets completed, the tasks get executed on the assigned nodes

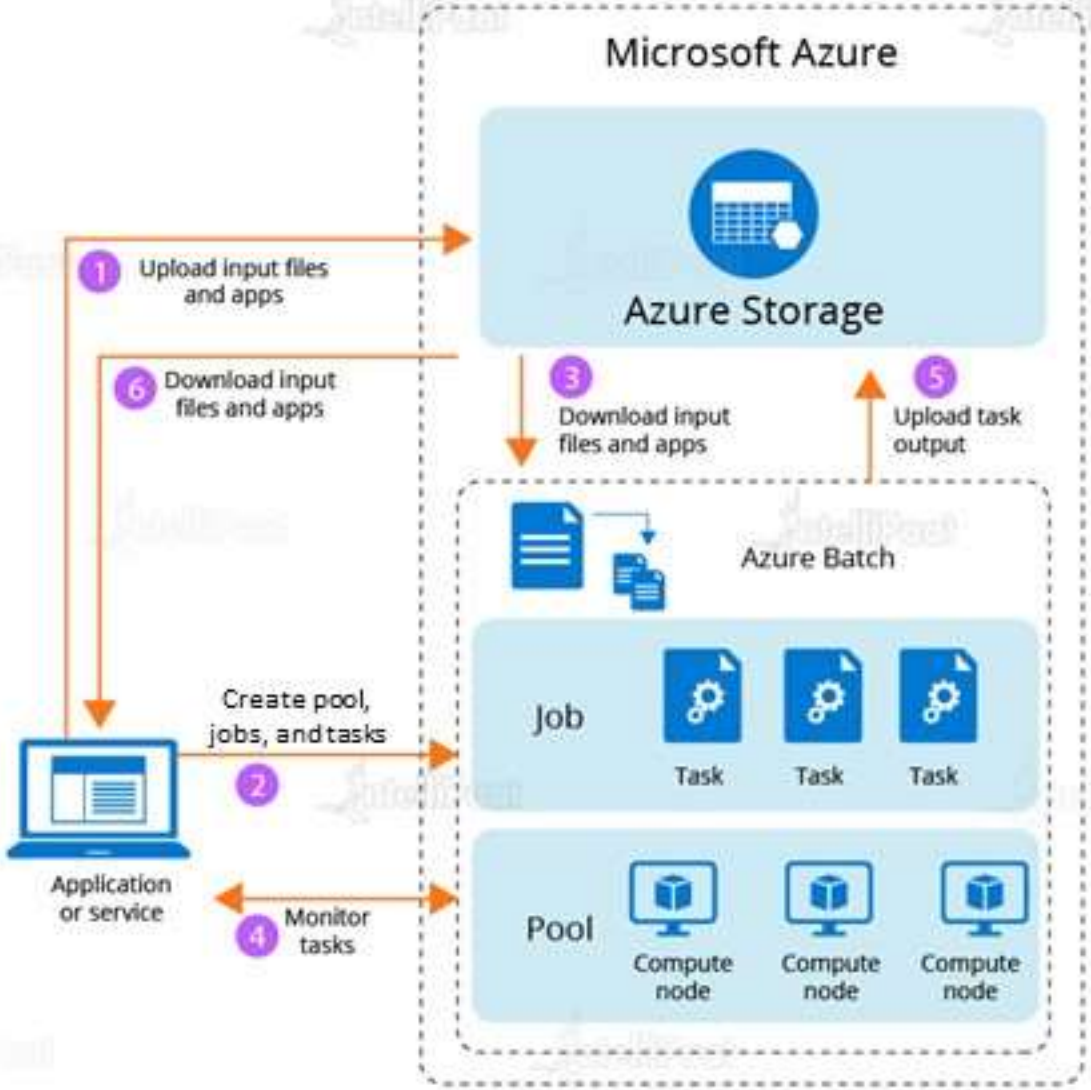
Working of Azure Batch



Step 4: Monitor task execution

- As the tasks run, query Batch to monitor the progress of each job and its tasks
- Our client application or service communicates with the Batch service over HTTPS

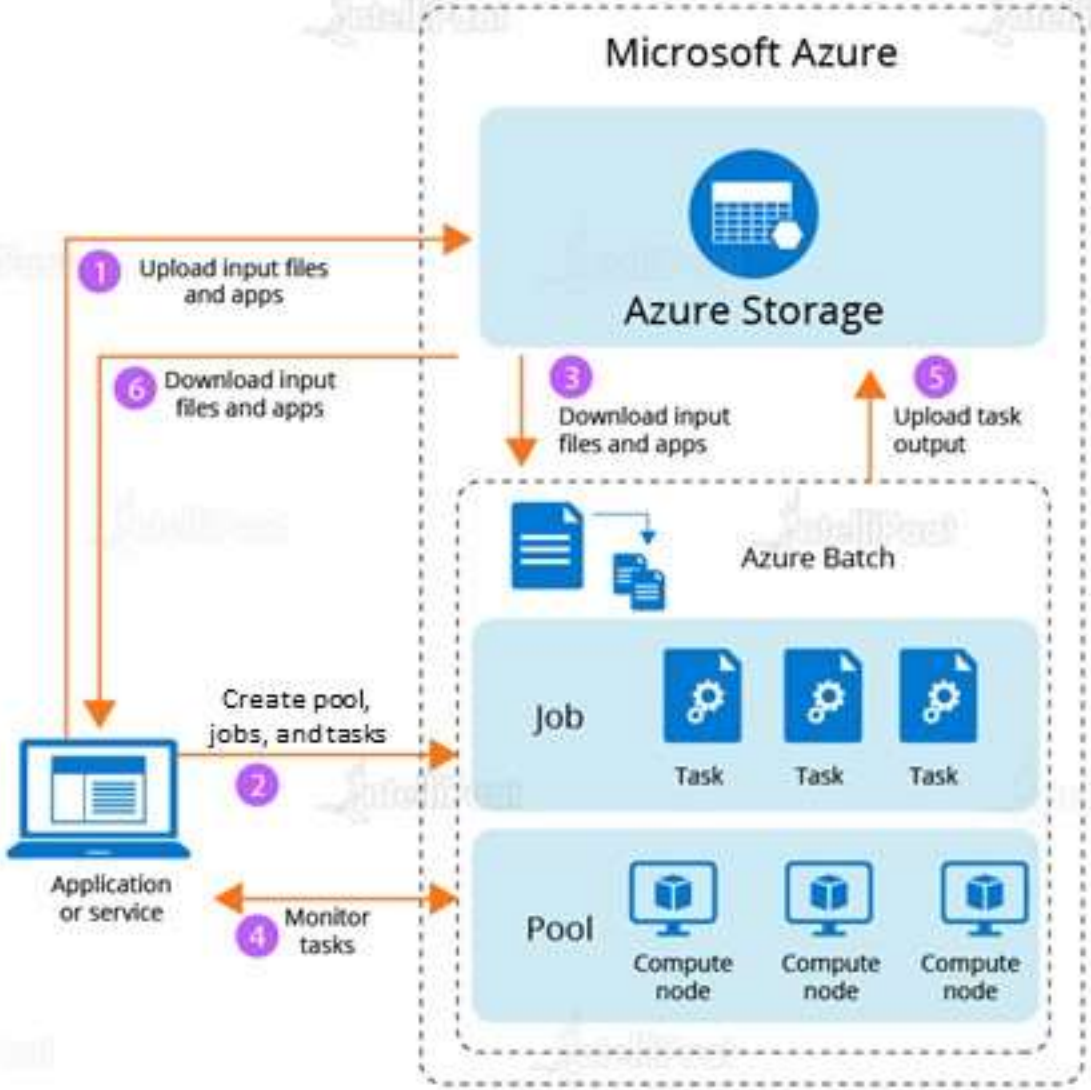
Working of Azure Batch



Step 5: Upload the task output

- As the tasks get completed, they can upload their result data to Azure Storage
- We can retrieve the files directly from the filesystem on a compute node

Working of Azure Batch



Step 6: Download output files

- When monitoring detects that the tasks in our job are completed, our client application or service can download the output data for further processing

Hands-on: Running a Batch Job Using Azure Portal

Hands-on: Parallel File Processing with Azure Batch Using the .NET API

Hands-on: Rendering a Blender Scene Using Batch Explorer

Hands-on: Parallel R Simulation with Azure Batch

Azure Data Factory

Azure Data Factory

Azure Data Factory falls under the identity domain of services in the SEO (Search Engine Optimization) catalog, and it is a cloud-based integration service

Data generated by several applications and products is increasing exponentially

Basically, it works on data. It orchestrates and automates the movement or transformation of data

Azure Data Factory

As data is coming from a number of different products, to analyze and store all this data, we need a powerful. Azure Data Factory helps us here by:

01 Storing data with the help of Azure Data Lake storage

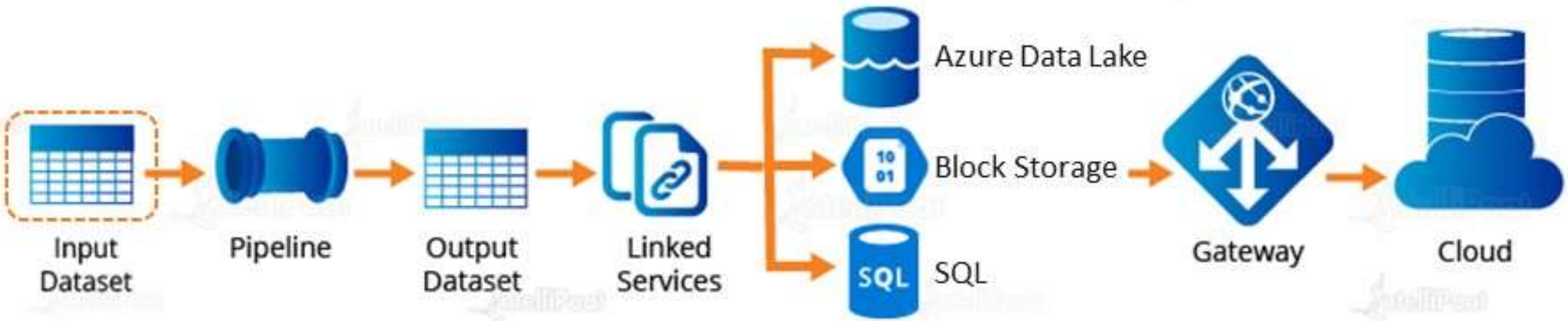
02 Analyzing the data

03 Transforming the data with the help of pipelines (logical grouping of activities that together perform a task)

04 Publishing the organized data

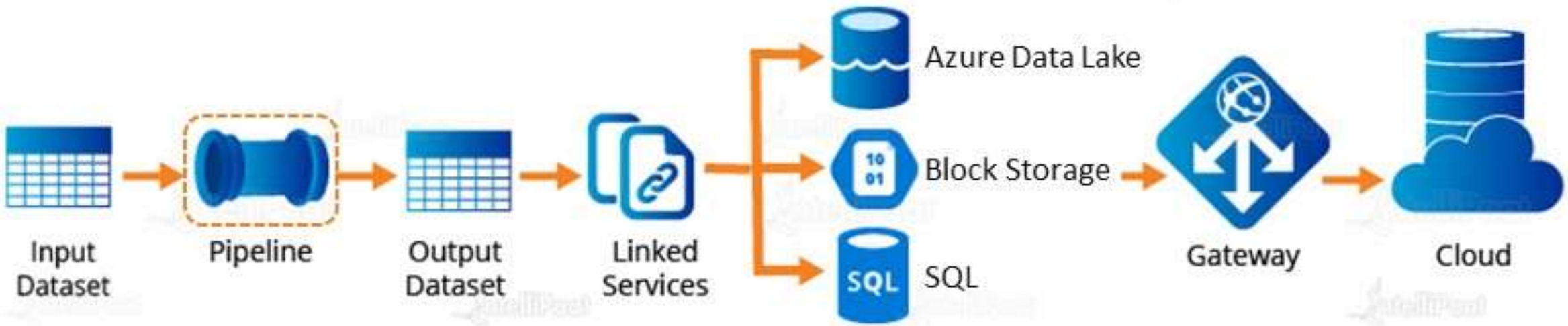
05 Visualizing the data with third-party applications like Apache Spark/Hadoop

Flow Process of Data Factory



01 **Input Dataset:** Data that we have within our data store. We need it to be processed, so we pass this data through a pipeline

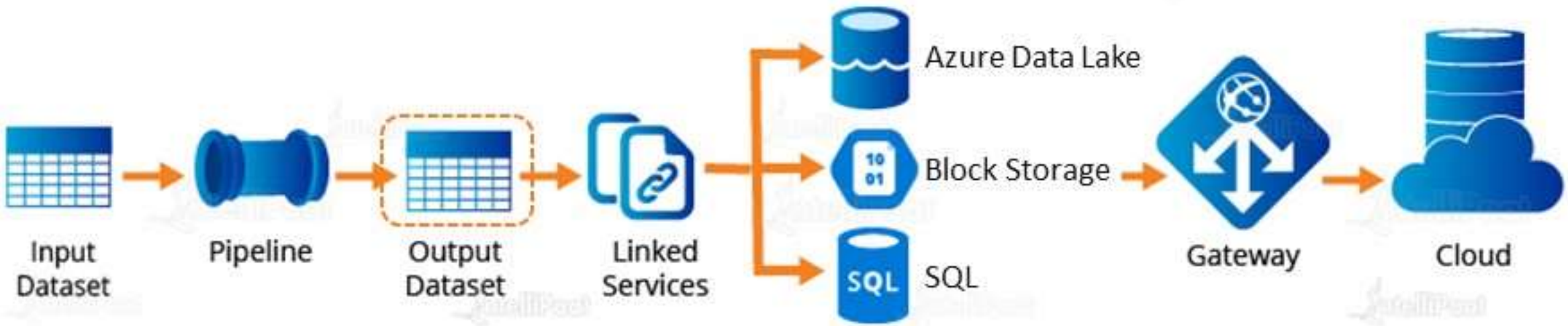
Flow Process of Data Factory



02

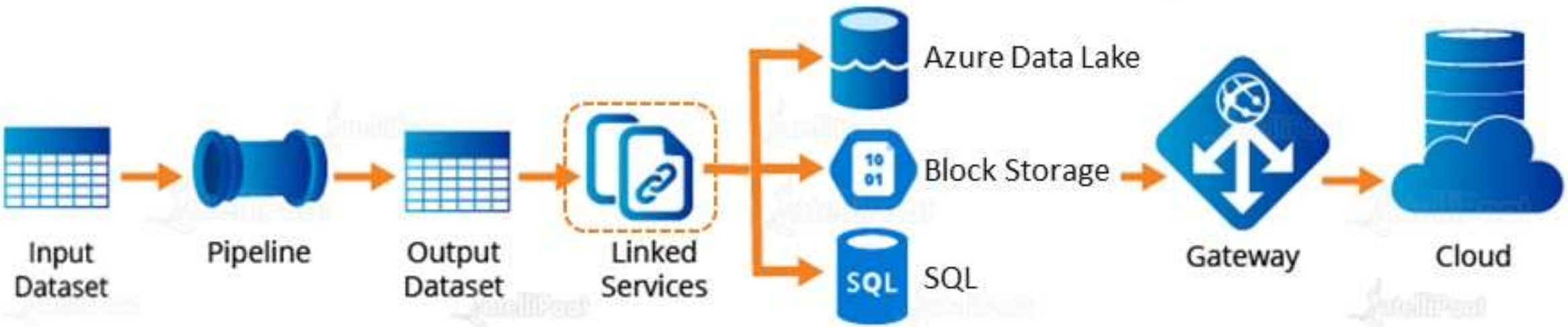
Pipeline: It basically performs an operation on the data that transforms data, which could be anything from just data movement to some data transformation

Flow Process of Data Factory



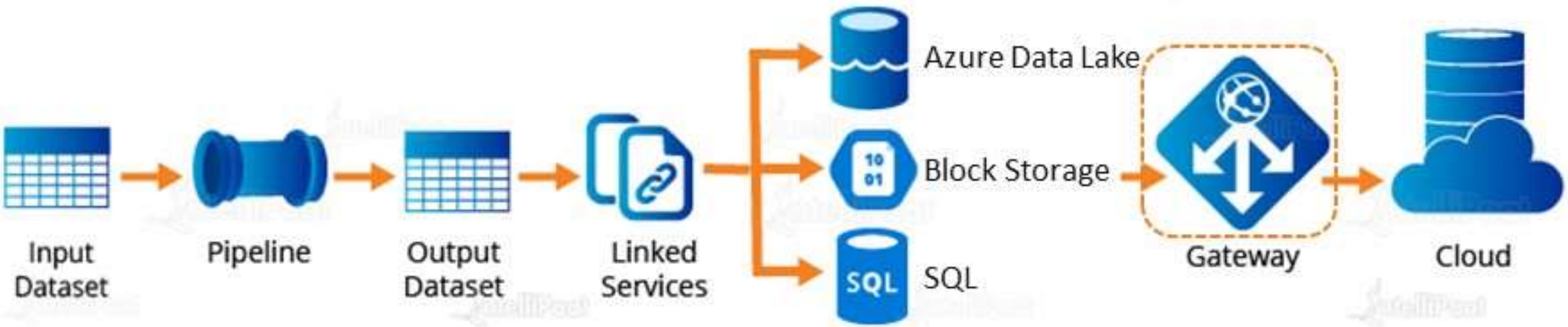
03 **Output Dataset:** It contains the data that is in a structured format as it has already been transformed in the pipeline storage. It is, then. given to linked services such as Azure Data Lake, Azure Blob Storage, or SQL

Flow Process of Data Factory



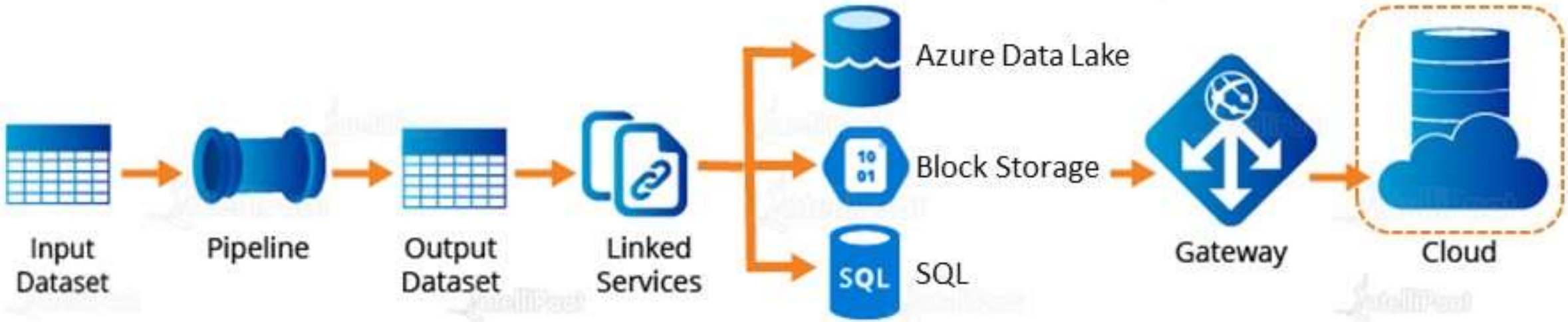
04 **Linked Services:** These store information that is very important when it comes to connecting to an external source

Flow Process of Data Factory



05 **Gateway:** It connects our on-premises data to the cloud. So, we do need a client installed on our on-premises system so that we can connect to Azure cloud

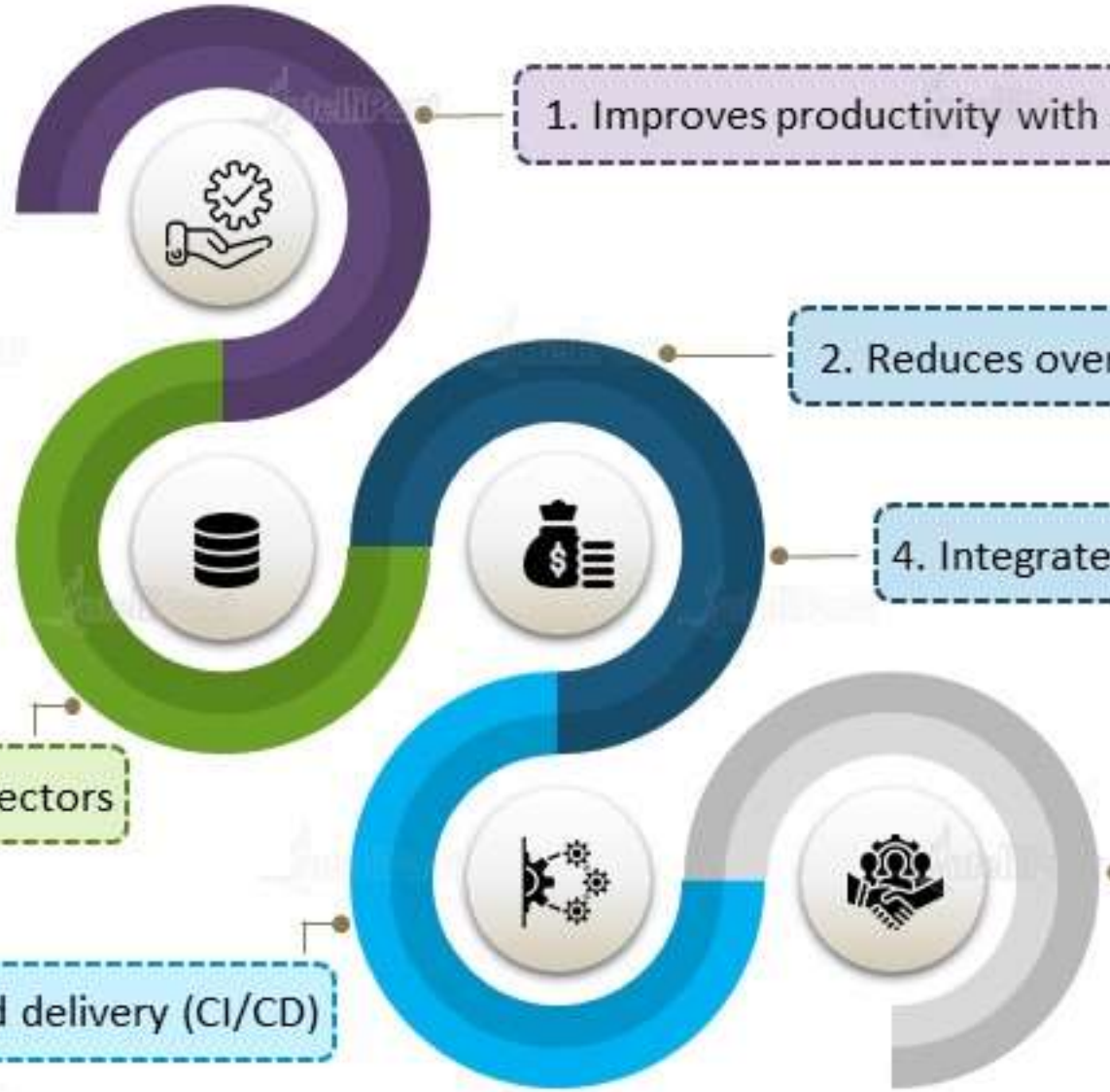
Flow Process of Data Factory



06 **Cloud:** Our data can be analyzed and visualized with a number of different analytical software such as Apache Spark, R, Hadoop, and so on

Why Azure Data Factory?

Why Azure Data Factory?



1. Improves productivity with shorter Time to Market
2. Reduces overhead costs
3. Transfers data using prebuilt connectors
4. Integrates data cost-effectively
5. Works the way we want
6. Gets continuous integration and delivery (CI/CD)

Why Azure Data Factory?

1. Improves productivity with shorter Time to Market

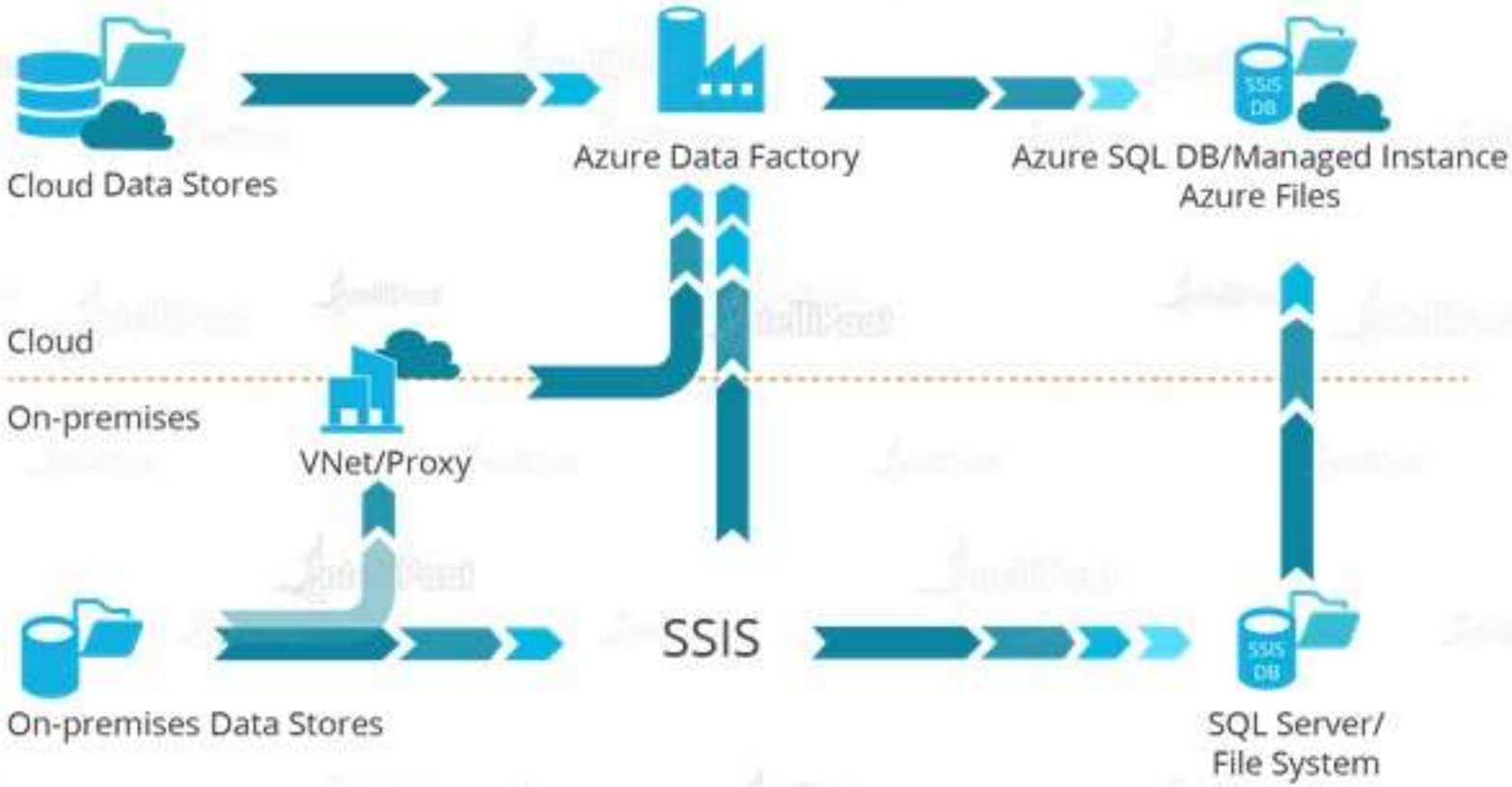
It develops simple and comprehensive ETL and ELT processes without coding or maintenance

It ingests, moves, prepares, transforms, and processes our data in a few clicks and completes our data modeling within the accessible visual environment

The managed Apache Spark™ service takes care of code generation and maintenance

Why Azure Data Factory?

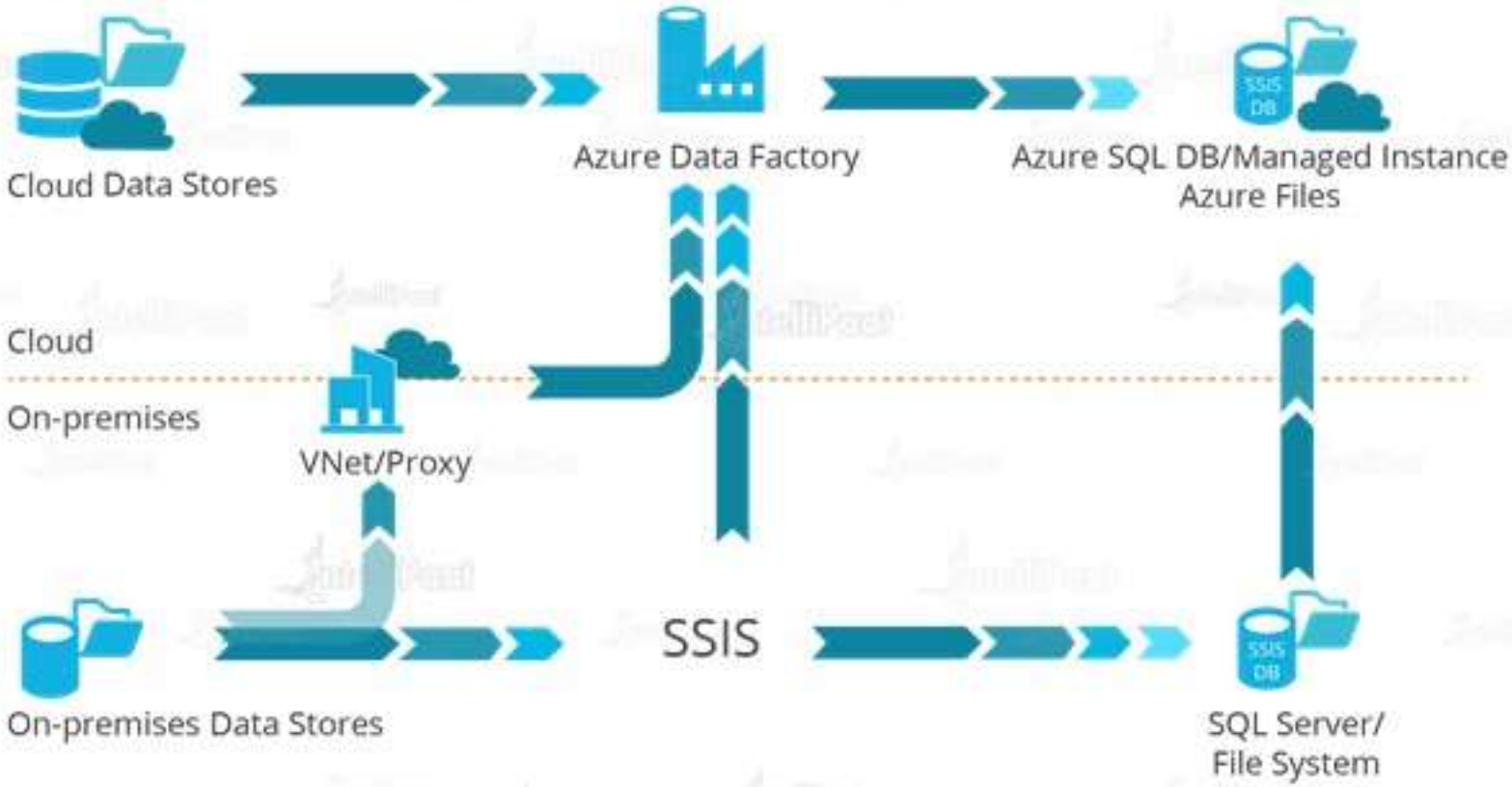
2. Reduces overhead costs



When migrating our SQL Server DB to the cloud, it preserves our ETL processes and reduces operational costs and complexity

Why Azure Data Factory?

2. Reduces overhead costs



It re-hosts on-premises SSIS packages in the cloud with minimal effort using Azure SSIS integration runtime. ETL in Azure Data Factory provides us with familiar SSIS tools

Why Azure Data Factory?

3. Transfers data using prebuilt connectors

It gives access to the ever-expanding portfolio of 90+ prebuilt connectors including Azure data services, on-premises data sources, Amazon S3 and Redshift, and Google BigQuery at no additional cost

Why Azure Data Factory?

4. Integrates data cost-effectively

It integrates our data using a serverless tool with no infrastructure to manage

For Azure Data Factory, we need to pay only for what we use, and we can scale up the elastic capabilities as our data grows

It transforms data with speed and scalability using the Apache Spark engine in Azure Databricks

Why Azure Data Factory?

5. Works the way we want

Azure Data Factory provides a single hybrid data integration service for all skill levels

We can use the visual interface or write our own code in Python, .NET, or ARM to build pipelines

We can add any processing service into the managed data pipelines or insert custom code as a processing step in any pipeline

Why Azure Data Factory?

6. Gets continuous integration and delivery (CI/CD)

It continuously monitors and manages pipeline performance, alongside applications, from a single console with Azure Monitor

It integrates our DevOps processes using the built-in support for pipeline monitoring

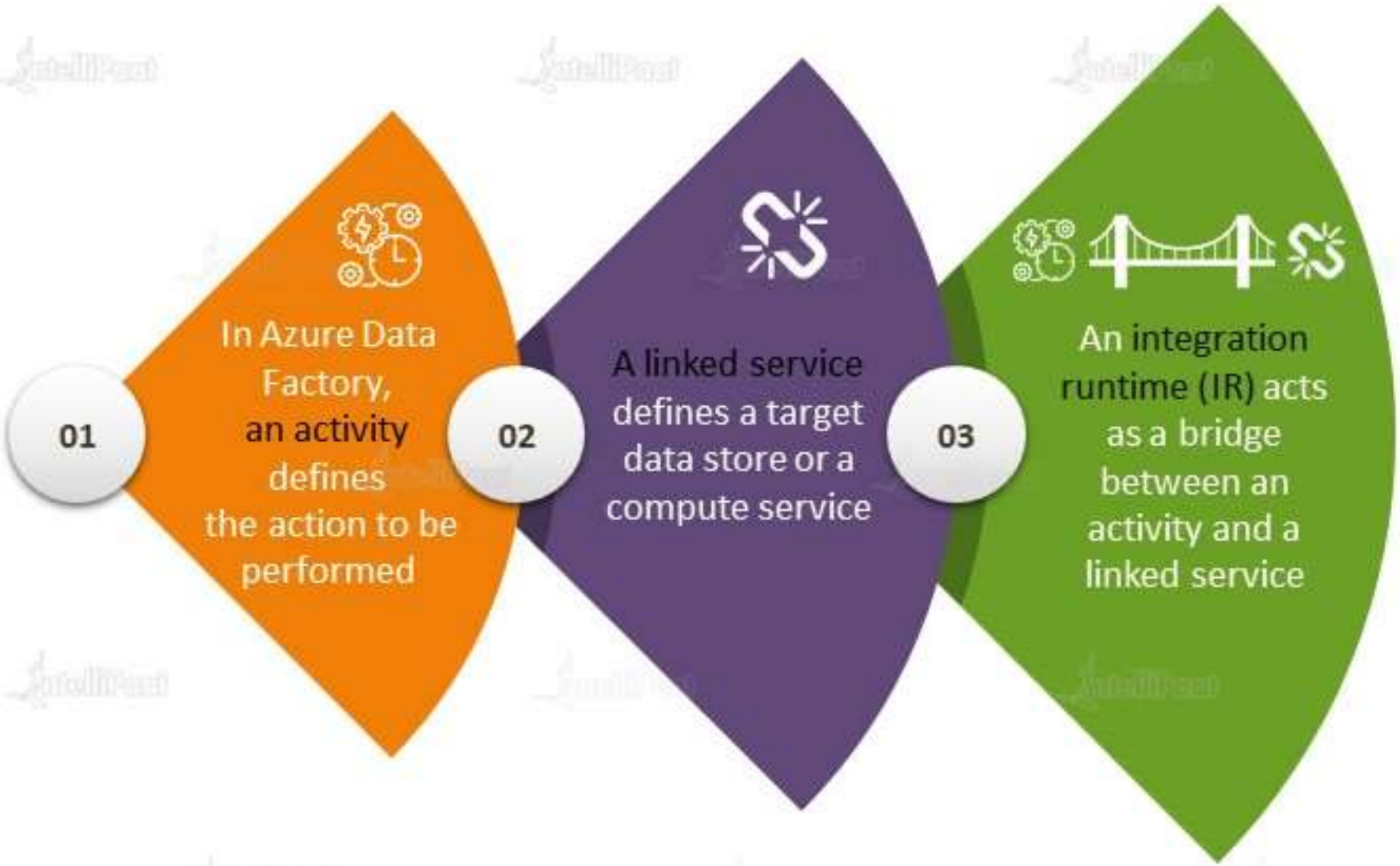
If we prefer a less programmatic approach, we can use the built-in visual monitoring tools and alerts

Integration Runtime in Azure Data Factory

Integration Runtime in Azure Data Factory

Integration Runtime (IR) is the compute infrastructure used by Azure Data Factory to provide following data integration capabilities across different network environments:

Integration Runtime in Azure Data Factory

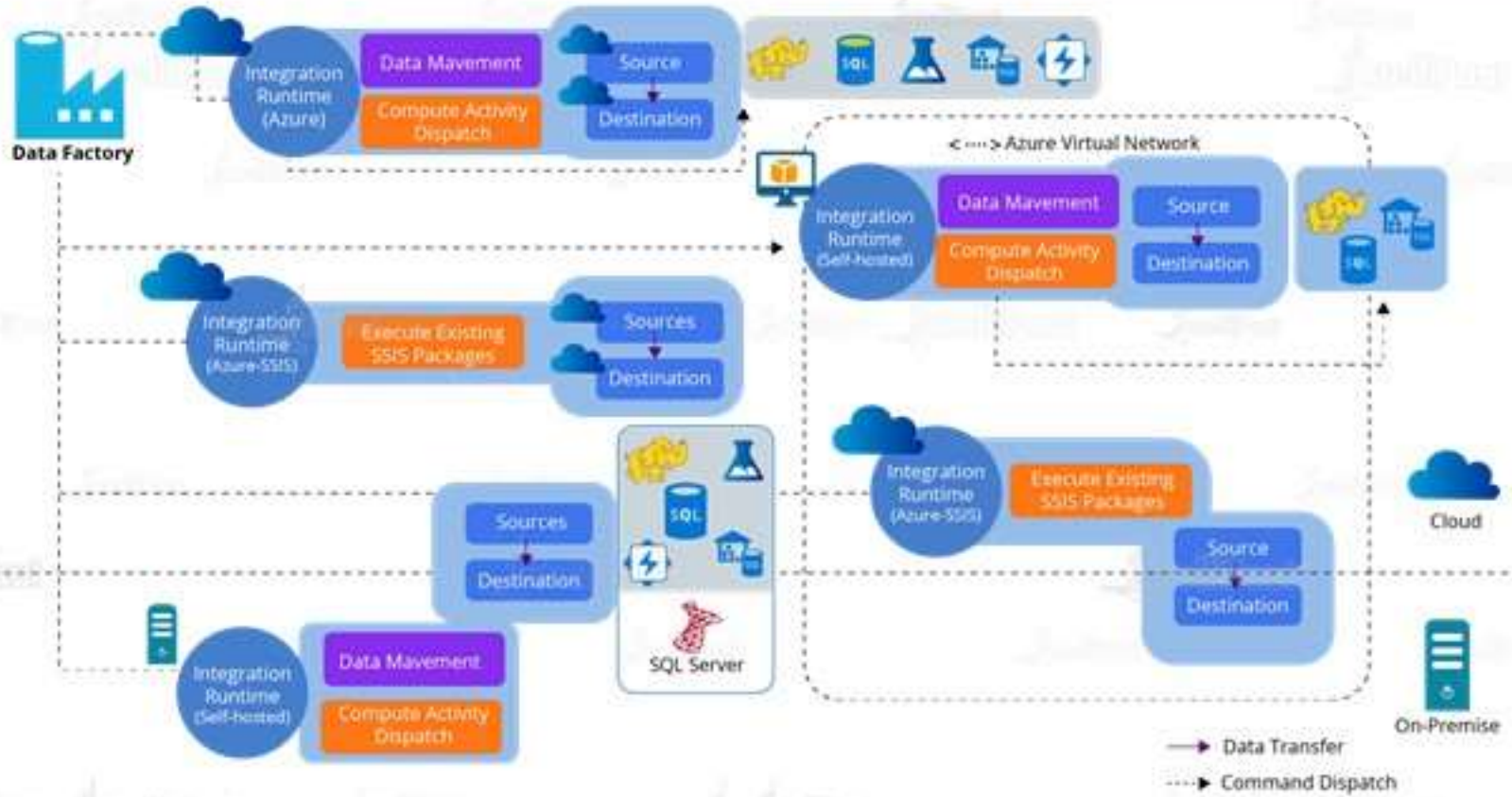


Integration Runtime Types in Azure Data Factory

Azure Data Factory offers three types of Integration Runtime, and we can choose the type that best fits the data integration capabilities and network environments we are looking for. These three types are:

Integration Runtime Types in Azure Data Factory

The diagram shows how different integration runtimes can be used in combination to offer rich data integration capabilities and network support:



Integration Runtime Location

Meanwhile, a Data Factory can access data stores and compute services in other Azure regions to move data between data stores or process data using compute services

Data Factory location is where the metadata of the Data Factory is stored and where the triggering of the pipeline is initiated from

This behaviour is realized through the **globally available IR** to ensure data compliance, efficiency, and reduced network egress costs

Integration Runtime Location

The IR location defines the location of its backend compute and, essentially, the location wherein data movement, activity dispatching, and SSIS package execution are performed

The IR location can be different from the location of the Data Factory it belongs to

Integration Runtime Types in Azure Data Factory

- Azure Integration Runtime
- Self-hosted Integration Runtime
- Azure-SSIS Integration Runtime

An Azure Integration Runtime is capable of:

Running Data Flows in Azure

Running copy activity between cloud data stores

Dispatching the following transform activities in a public network:

Databricks Notebook/Jar/ Python activity

HDInsight Hive activity

Machine Learning Batch Execution activity

... and many more

Integration Runtime Types in Azure Data Factory

- Azure Integration Runtime
- Self-hosted Integration Runtime
- Azure-SSIS Integration Runtime

Azure IR Network Environment

- Azure Integration Runtime supports connecting to data stores and compute services with public accessible endpoints

Integration Runtime Types in Azure Data Factory

IntelliPaat

- Azure Integration Runtime
- Self-hosted Integration Runtime
- Azure-SSIS Integration Runtime

Azure IR Compute Resource and Scaling

- Azure integration runtime provides a fully managed, serverless compute in Azure
- It provides native compute to move data between cloud data stores in a secure, reliable, and high-performance manner

Integration Runtime Types in Azure Data Factory

- **Azure Integration Runtime**
- Self-hosted Integration Runtime
- Azure-SSIS Integration Runtime

Azure IR Location

We can set a certain location of an Azure IR, in which case the data movement or activity dispatch will happen in that specific region

Integration Runtime Types in Azure Data Factory

IntelliPaat

- Azure Integration Runtime
- **Self-hosted Integration Runtime**
- Azure-SSIS Integration Runtime

A self-hosted IR is capable of:

Running the copy activity between a cloud data store and a data store in a private network

Dispatching the following transform activities against compute resources in on-premises or in Azure Virtual Network:

- Data Lake Analytics U-SQL activity
- HDInsight Hive activity (BYOC)
- Machine Learning Batch Execution activity

... and many more

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

Azure-SSIS Integration Runtime

Self-hosted IR Network Environment

- If we want to perform data integration securely in a private network environment, which does not have a direct line-of-sight from the public cloud environment, we can install a self-hosted IR on the on-premises environment behind our corporate firewall, or inside a virtual private network
- The self-hosted integration runtime only makes outbound HTTP-based connections to open the Internet

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

Azure-SSIS Integration Runtime

Self-hosted IR Compute Resource and Scaling

- Self-hosted IR needs to be installed on an on-premises machine or a virtual machine inside a private network
- Currently, running a self-hosted IR is only supported on the Windows operating system
- For high-availability and scalability, we can scale up the self-hosted IR by associating the logical instance with multiple on-premises machines in an active–active mode

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

Azure-SSIS Integration Runtime

Self-hosted IR Location

- The self-hosted IR is logically registered to the Data Factory and the compute used to support its functionalities is provided by us
- Therefore, there is no explicit location property for a self-hosted IR
- When used to perform data movement, the self-hosted IR extracts data from the source and writes it into the destination

Integration Runtime Types in Azure Data Factory

- Azure Integration Runtime
- Self-hosted Integration Runtime
- Azure-SSIS Integration Runtime

To lift and shift the existing SSIS workload, we can create an **Azure-SSIS IR** to natively execute SSIS packages

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

Azure-SSIS Integration Runtime

Azure-SSIS IR Network Environment

- Azure-SSIS IR can be provisioned in either a public network or a private network
- On-premises data access is supported by joining Azure-SSIS IR to a virtual network that is connected to our on-premises network

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

Azure-SSIS Integration Runtime

Azure-SSIS IR Compute Resource and Scaling

- Azure-SSIS IR is a fully managed cluster of Azure VMs dedicated to run our SSIS packages
- We can bring our own Azure SQL Database or the managed instance server to host the catalog of SSIS projects/packages (SSISDB) that is going to be attached to it

Integration Runtime Types in Azure Data Factory

Azure Integration Runtime

Self-hosted Integration Runtime

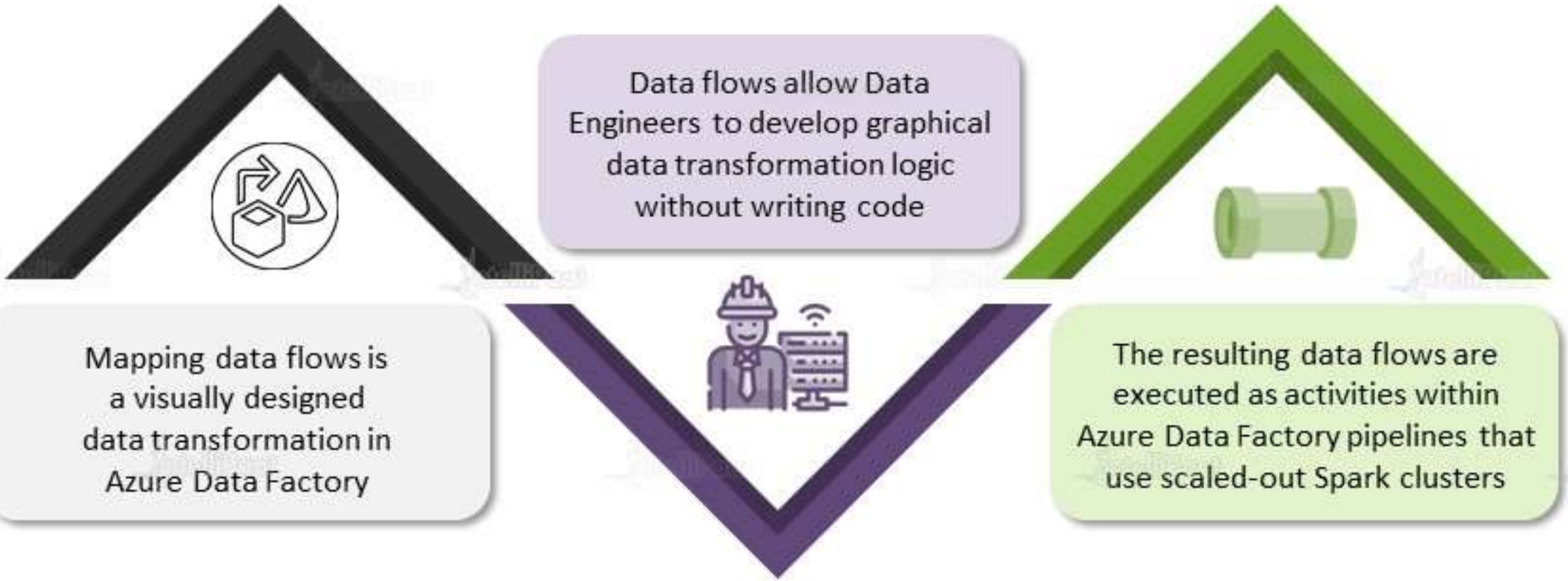
Azure-SSIS Integration Runtime

Azure-SSIS IR Compute Resource and Scaling

- We can scale up the power of compute by specifying the node size and specifying the number of nodes in a cluster
- We can manage the cost of running our Azure-SSIS Integration Runtime by stopping and starting it as per our requirement

Mapping Data Flows

Mapping Data Flows



Mapping Data Flows

Hands-on: Transforming Data Using Mapping Data Flows

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor