Pipeline: A data integration workload unit in Azure Data Factory. A logical grouping of activities assembled to execute a particular data integration process.

• **Activity**: Performs a task inside a pipeline, for example, copying data from one place to another.

• **Dataset**: Contains metadata describing a specific set of data held in an external storage system. Pipeline activities use datasets to interact with external data.

• **Linked service:** Represents a connection to an external storage system or external compute resource. • Integration runtime: Provides access to internal compute resource inside Azure Data Factory. ADF has no internal storage resources.

• **Debug**: You can run a pipeline interactively from the ADF UX using "Debug" mode. This means that the pipeline definition from the ADF UX session is executed – it does not need to be published to the connected factory instance. During a debugging run, a pipeline treats external resources in exactly the same way as in published pipeline runs.

• **Copy Data tool:** A wizard-style experience in the ADF UX that creates a pipeline to copy data from one place to another, but in practice you are unlikely to use the tool very often.

• **Azure Storage:** Microsoft's cloud-based managed storage platform.

• **Storage account**: A storage account is created in order to use Azure Storage services.

• **Storage key**: Storage keys are tokens used to authorize access to a storage account. You can manage an account's keys in the Azure portal.

• **Blob storage**: General-purpose file (blob) storage, one of the types of storage offered by Azure Storage. Other supported storage types (not described here) include file shares, queues, and tables.

• **Container**: Files in blob storage are stored in containers, subdivisions of a storage account's blob storage. Blob storage is divided into containers at the root level only – they cannot be nested.

• **Azure Storage Explorer**: An app used to manage Azure Storage accounts, available online and as a desktop application.

• **Bandwidth**: A term used by Microsoft to describe the movement of data into and out of Azure data centers. Outbound data movements incur a fee, sometimes referred to as an egress charge.

• **Unstructured file**: A file treated as having no internal data structure – a blob. The Copy data activity treats files as unstructured when a binary copy is specified.

• **Structured file**: A file with a tabular data structure such as CSV or Parquet.

• **Parquet file**: A column-oriented, compressed structured file format supporting efficient storage and querying of large volumes of data.

• **Semi-structured file**: A file with a nontabular, frequently nested data structure, such as XML or JSON. • **Collection reference**: Nested data structures can represent multiple collections of data simultaneously. In a Copy data activity schema mapping, the collection reference indicates which of the collections is being transformed.

• **Sink**: Azure Data Factory refers to data pipeline destinations as sinks.

• **Interim data type**: The Copy data activity converts incoming data values from their source types to interim ADF data types, then converts them to corresponding sink system type. This makes extensibility of ADF to support new datasets easier and faster.

• **Data integration unit (DIU)**: A DIU is a measure of computing power incorporating CPU, memory, and network usage. Power is allocated to Copy data activity executions as a number of DIUs; the cost of an execution is determined by the duration for which it was allocated those DIUs.

• **Degree of parallelism (DoP)**: A Copy data activity can be performed in parallel using multiple threads to read different files simultaneously. The maximum number of threads used during an activity's execution is its degree of parallelism; the number can be set manually for the activity, but this is not advised.

• **Azure SQL DB**: Azure-based, PaaS SQL Server service.

• **Logical SQL Server**: Logical grouping of Azure SQL Databases for collective management.

• **Online query editor**: Web-based query editor available for use with Azure SQL DB (and other Azure database platforms)

• **Expression**: An expression is evaluated at pipeline execution time to determine a property value. The data type of an expression is string, integer, float, boolean, array, or dictionary.

• **Array**: A collection of multiple values referred to as elements. Elements are addressed by an integer index between zero and one less than the array's length.

• **Dictionary**: A collection whose elements are referred to by name.

• **Expression builder**: An expression editor built into the ADF UX.

• **System variable**: System variables provide access to the runtime values of various system properties.

• **User variable:** Created to store String, Boolean, or Array values during a pipeline's execution.

• **Expression function**: One of a library of functions available for use in expressions. Function types include String, Math, Logical, Date, Collection, and Type conversions.

• **Interpolated string**: A string literal containing placeholder expressions.

• **Placeholder expression**: An expression embedded in an interpolated string, evaluated at runtime to return a string.

• **Escape**: String literals beginning with the @ character must be escaped to prevent their interpretation as expressions. @ is escaped by following it with a second @ character.

• **Stored procedure activity**: ADF activity that enables the execution of a database stored procedure, specifying values for the stored procedure's parameters as required.

• **Lookup activity**: ADF activity that returns one or more rows from a dataset for use during a pipeline's execution. In the case of SQL Server datasets, rows can be returned from tables, views, inline queries, or stored procedures.

• **Set variable activity**: ADF activity used to update the value of a user variable.

• **Append variable activity**: ADF activity used to add a new element to the end of an existing Array variable's value.

• **Activity dependency**: Constraint used to control the order in which a pipeline's activities are executed.

• **Activity output object**: JSON object produced by the execution of an ADF activity. An output object and its properties are available to any activity dependent on the source activity, either directly or indirectly.

• **Breakpoint**: An ADF UX breakpoint allows you to run a pipeline, in debug mode, up to and including the activity on which the breakpoint is set (breakpoints do not exist in published pipelines). Unlike in other IDEs, it is not possible to resume execution after hitting a breakpoint.

• **$$FILEPATH**: A reserved system variable that enables the Copy data activity to label incoming file data with its source file. $$FILEPATH is available solely to populate additional columns in the Copy data activity and cannot be used in expressions.

• **$$COLUMN**: A reserved system variable that enables the Copy data activity to duplicate a specified column in incoming data. $$COLUMN is available solely to populate additional columns in the Copy data activity and cannot be used in expressions.

• **Additional columns**: A Copy data activity source can be augmented with additional columns, the values of which are specified by an expression, $$FILEPATH, $$COLUMN, or a hard-coded static value.

• **Lineage tracking**: The practice of labeling data as it is processed, to enable later identification of information related to its source and/or processing.

• **Runtime parameter**: A placeholder in a factory resource definition for a value substituted at runtime. Pipeline, dataset, and linked service parameters are runtime parameters; global parameters are not.

• **Optional parameter**: A runtime parameter can be made optional by defining a default value for it. If a value is not supplied at runtime, the default value is substituted for the parameter instead.

• **Reusability**: Runtime parameters enable factory resources to be defined in a reusable way, using different parameter values to vary resource behavior.

• **Global parameter**: A constant value, shared by all pipelines, not expected to change frequently. Global parameters are referred to in ADF expressions, within the scope of pipeline activities, using the syntax pipeline().globalParameters.ParameterName.

• **Pipeline parameter**: Runtime parameter for an ADF pipeline. Pipeline parameters are referred to in ADF expressions, within the scope of pipeline activities, using the syntax pipeline(). parameters.ParameterName.

• **Dataset parameter**: Runtime parameter for an ADF dataset. Dataset parameters are referred to in ADF expressions, within the scope of the dataset, using the syntax dataset().ParameterName.

• **Linked service parameter**: Runtime parameter for an ADF linked service. Dataset parameters are referred to in ADF expressions, within the scope of the linked service, using the syntax linkedService().ParameterName. The ADF UX does not always support parameter management for linked services, but parameters can be defined and used by editing a linked service's JSON definition.

• **Execute Pipeline activity**: ADF pipeline activity used to execute another pipeline within the same data factory instance.

• **Azure Key Vault**: A secure repository for secrets and cryptographic keys.

• **Secret**: A name/value pair stored in an Azure Key Vault. The value usually contains sensitive information such as service authentication credentials. A secure way to handle this information is to refer to the secret by name – a service that requires the secret's value may retrieve it from the vault by name if permitted to do so.

• **Service principal**: An identity created for use with an application or service – such as Azure Data Factory – enabling the service to be identified when external resources require authentication and authorization.

• **Managed identity**: A managed identity associates a service principal with an instance of Azure Data Factory (or other Azure resources). A system-assigned managed identity is created automatically for new ADF instances created in the Azure portal and is automatically removed if and when its factory is deleted.

• **Access policy**: A key vault's access policy defines which service or user principals are permitted to access data stored in the vault.

• **Dependency condition**: Characterizes an activity dependency. An activity dependent on another is only executed if the associated dependency condition is met – that is, depending on whether the prior activity succeeds, fails, or is skipped.

• **Multiple dependencies**: An activity dependent on multiple activities is only executed if each prior activity satisfies a dependency condition. If an activity specifies multiple dependency conditions on the same prior activity, only one needs to be met.

• **Leaf activity**: A leaf activity is a pipeline activity with no successors.

• **Conditional activities**: ADF has two conditional activities – the If Condition activity and the Switch activity.

• **If Condition activity**: Specifies an expression that evaluates to true or false and two corresponding contained sets of activities. When the activity runs, its expression is evaluated, and the corresponding activity set is executed.

• **Switch activity**: Specifies an expression that evaluates to a string value and up to 25 corresponding contained sets of activities. When the activity runs, the expression is evaluated, and the corresponding activity set, if any, is executed. If no matching case is found, the default activity set is executed instead.

• **Iteration activities**: ADF has two iteration activities – the ForEach activity and the Until activity.

• **ForEach activity**: Specifies a JSON array and a set of activities to be executed once for each element of the array. The current element of the array is addressed in each iteration using the item() expression.

• **Parallelism**: By default, ForEach activity executions take place in parallel, requiring care to ensure that activities from simultaneous iterations do not interfere with one another. Variable modification must be avoided, but the Execute Pipeline activity provides an easy way to isolate iterations. The ADF UX executes ForEach iterations sequentially in Debug mode, which can make parallelism faults hard to detect at development time.

• **Until activity**: Specifies a terminating condition and a set of activities to be executed repeatedly until the terminating condition is met. Activities within an Until activity are executed at least once and never in parallel.

• **Nesting**: Iteration activities may not be nested in other iteration activities. Conditional activities may not be nested in other conditional activities, although nesting in iteration activities is permitted. A common workaround is to implement inner and outer activities in separate pipelines, calling the inner activity from the outer via the Execute Pipeline activity.

• **Breakpoints**: The ADF UX does not support breakpoints inside iteration or conditional activities.

• **Get Metadata activity:** Returns metadata that describes attributes of an ADF dataset. Not all metadata attributes are supported by all datasets – nonexistent dataset targets will cause errors unless the "Exists" argument is specified; specifying the "Child Items" argument on a nonfolder target will cause the activity to fail.

• **Fault tolerance**: The Copy data activity supports enhanced error handling through its Fault tolerance settings, enabling individual error rows to be diverted into an external log file without completely abandoning a data load.

• **Raising errors**: At the time of writing, ADF has no "raise error" activity. Approaches available to manufacture errors include defining illegal type casts in Set variable activities or exploiting error raising functionality in external services, for example, by using SQL Server's RAISERROR or THROW statements.

•**Apache Spark**: Open source data processing engine that automatically distributes processing workloads across a cluster of servers (referred to as nodes) to enable highly parallelized execution.

• **Databricks**: Data processing and analytics platform built on Apache Spark and adding a variety of enterprise features.

• **Data flows**: ADF's visual data transformation tool, built on Azure Databricks.

• **Data flow debug**: Data flow debug mode provisions a Databricks cluster on which you can execute data flows from the ADF UX.

• **Time to live** (TTL): The data flow debug cluster has a default TTL of one hour, after which – if it is not being used – it automatically shuts down.

• **Data flow activity**: ADF pipeline activity used to execute a data flow.

• **Parameters**: Data flow parameters are specified in Debug Settings during development and substituted for values supplied by the calling Data flow activity at runtime.

• **Data flow canvas**: Visual development environment for data flows.

• **Transformation**: A data flow is made up of a sequence of connected transformations, each of which modifies a data stream in some way.

• **Output stream name**: Name that uniquely identifies each transformation in a data flow.

• Inspect tab: Use a transformation's Inspect tab to view input and output schema information.

• **Data preview tab**: Use a transformation's Data preview tab to preview data emitted by the transformation. Using data preview requires data flow debug to be enabled and can be used to delay an approaching cluster timeout.

• **Optimize tab**: Use a transformation's Optimize tab to influence data partitioning in Spark when the transformation is executed.

• **Source transformation**: Reads input data from an external source. Every data flow starts with one or more Source transformations.

• **Sink transformation**: Write transformed data to an external source. Every data flow ends with one or more Sink transformations.

• **Data flow expression language**: Data flow expressions have their own language and expression builder, different from those of ADF pipeline expressions.

• **Data Flow Script**: Language in which data flow transformations are stored, embedded in a data flow's JSON file.

• **Column patterns**: Where supported, use column patterns to specify multiple columns which are to be handled in the same way. Columns are specified using data flow expressions to match column metadata.

• **Filter transformation**: Selects rows from its input data stream to be included in its output stream, on the basis of criteria specified as a data flow expression. Other rows are discarded.

• **Lookup transformation**: Conceptually similar to a SQL join between two data streams. Supports a variety of join styles and criteria.

• **Derived Column transformation**: Uses data flow expressions to derive new columns for inclusion in a data flow.

• **Locals**: Named intermediate derivations in a Derived Column transformation. Used to simplify expressions and eliminate redundancy.

• **Select transformation**: Used to rename columns in a data flow or to remove them.

• **Aggregate transformation**: Aggregates one or more columns in a data flow, optionally grouping by other specified columns.

• **Exists transformation**: Selects rows from its input data stream to be included in its output stream, on the basis of the existence (or not) of matching rows in a second data stream. Other rows are discarded.

• **Templates**: Reusable implementations of common pipeline and data flow patterns.

• **Template gallery**: Source of provided templates, accessed using the Create pipeline from template bubble on the Data Factory overview page.

•**External pipeline activity**: An ADF pipeline activity executed using compute resource provided by a service outside Azure Data Factory, for example, Stored procedure, Databricks, or HDInsight activities.

• **Internal pipeline activity**: An ADF pipeline activity executed using compute resource provided internally by Azure Data Factory.

• **Integration runtime**: Internal compute resource managed by Azure Data Factory.

• **Dispatching**: Management of ADF activity execution, particularly external pipeline activities.

• **Azure IR**: A fully managed, serverless integration runtime that executes data movements and transformations defined by the Copy data and Data flow activities. Azure IRs also manage dispatching of external activities to storage and compute environments like Azure blob storage, Azure SQL Database, and others.

• **AutoResolveIntegrationRuntime**: An Azure IR present in every data factory instance. The location of IR compute is determined automatically at runtime, and Databricks clusters created for Data flow activities using this IR have a TTL of zero – you can modify these characteristics by creating and using your own Azure IR.

• **Self-hosted integration runtime**: An IR installed on one or more servers provided by you in a private network. A self-hosted IR permits you to expose private resources to ADF, for example, source systems which are hosted on-premises or for which no native ADF connector is available.

• **Linked self-hosted IR**: A self-hosted IR is connected to exactly one Azure Data Factory instance. A common pattern used to enable access in other data factories is to share it, enabling other data factories to create linked self-hosted IRs that refer to the shared self[1]hosted IR.

• **Azure-SSIS IR**: A fully managed integration runtime that supports SSIS package execution in ADF. An Azure-SSIS IR consists of a VM cluster of a specified size and power – although the cluster is managed for you, its infrastructure is more visible than in serverless Azure IRs.

• **Web activity**: ADF pipeline activity supporting calls to REST API endpoints

•**Power Query**: Graphical data preparation tool available in a number of Microsoft products, including ADF Power Query activities, Excel, Power Platform dataflows, or Power BI.

• **Data wrangling**: Interactive exploration and preparation of datasets.

• **Mashup**: Data wrangling transformation implemented in Power Query.

• **M formula language**: Language used to express transformations built in Power Query. M expressions built using the graphical Power Query Editor are translated into Data Flow Script at runtime by Azure Data Factory, for execution in the same way as an ADF data flow.

• **Power Query activity**: ADF pipeline activity used to execute Power Query mashups implemented in the ADF UX.

•**Azure Resource Manager (ARM) template**: An ARM template is a JSON file that defines components of an Azure solution, for example, the contents of an Azure Data Factory instance.

• **Publish**: To run a pipeline independently of the ADF UX, it must be deployed into a data factory's published environment. Published pipelines are executed using triggers. published pipeline runs can be observed in the ADF UX monitoring experience.

• **Publish branch**: A nominated branch in a factory's Git repository, by default adf_publish. The publish branch contains ARM templates produced when publishing factory resources in the ADF UX.

• **Azure custom role**: A custom security role, built by assembling a required set of permissions, to provide security profiles not supported in the standard Azure role set.

• **Deployment parameters**: Specified in an ARM template, deployment parameters enable different values to be substituted at deployment time. In the case of ADF, this permits a single template to be used for deployments to multiple different data factories.

• **Parameterization template**: A development data factory's parameterization template specifies which factory resource properties should be made parameterizable using deployment parameters.

• **CI/CD**: Continuous integration and continuous delivery (CI/CD) is a development practice in which software changes are integrated into the main code base and deployed into production continuously.

• **Azure Pipelines**: Microsoft's cloud-based CI/CD pipeline service.

• **Data serialization language**: Human-readable language used to represent data structures in text for storage or transmission. XML, JSON, and YAML are examples of data serialization languages.

• **YAML**: Data serialization language with an indentation-based layout, used in Azure DevOps to define Azure DevOps pipelines. YAML pipeline files are stored under version control like any other code file.

• **Task**: Configurable process used in an Azure DevOps pipeline, for example, script, AzureResourceManagerTemplateDeployment@3, or AzurePowerShell@4.

• **Pipeline variable**: Variable defined for use in an Azure DevOps pipeline. Secret variables allow secret values to be specified in pipeline YAML without storing them in version control.

• **Service connection**: Represents a nominated AAD principal with the permissions required by an Azure DevOps pipeline.

• **Feature branch workflow**: A common Git workflow in which development work takes place in isolated feature branches.

• **Pull request**: A request to merge a feature branch back into the collaboration branch when feature development is complete.

• **Az.DataFactory**: PowerShell module providing cmdlets for interacting with Azure Data Factory.

•**Trigger**: A unit of processing that runs one or more ADF pipelines when certain execution conditions are met. A pipeline can be associated with – and run by – more than one trigger.

• **Trigger run**: A single execution of a trigger. If the trigger is associated with multiple pipelines, one trigger run starts multiple pipeline runs. The ADF UX monitoring experience reports trigger and pipeline runs separately.

• **Trigger start date**: The date and time from which a trigger is active.

• **Trigger end date**: The date and time after which a trigger is no longer active.

• **Recurrence pattern**: A simple time-based scheduling model, defined by a repeated interval after a given start date and time.

• **Schedule trigger**: A trigger whose execution condition is defined by a recurrence pattern based on the trigger's start date or using a wall clock schedule.

• **Event-based trigger**: A trigger whose execution condition is the creation or deletion of a file from Azure blob storage.

• **Resource provider**: Azure uses resource providers to create and manage resources. In order to use a resource in an Azure subscription, the corresponding resource provider must be registered to that subscription.

• **Azure Event Grid:** Cloud service providing infrastructure for event[1]driven architectures. Azure blob storage uses Event Grid to publish file creation and other events; ADF subscribes to Event Grid to consume events and run event-based triggers.

• **Tumbling window trigger**: A trigger that uses a recurrence pattern based on the trigger's start date to define a sequence of processing windows between successive executions. Tumbling window triggers also support more advanced scheduling behaviors like dependencies, concurrency limits, and retries.

• **Pipeline run overlap**: Pipeline runs may overlap if a trigger starts a new pipeline run before a previous one has finished. Use a tumbling window self-dependency with a concurrency limit of one to prevent this.

• **Reusable triggers**: A single schedule or event-based trigger can be used by multiple pipelines. A tumbling window trigger can be used by a single pipeline

. • **Trigger-scoped system variables**: ADF system variables available for use in trigger definitions. Some trigger-scoped variables are specific to the type of ADF trigger in use.

• **Azure Logic Apps**: Cloud service for general-purpose task scheduling, orchestration, and automation. Internally, ADF triggers are implemented using Azure Logic Apps.

• **Trigger publishing:** Triggers do not operate in the ADF UX debugging environment and must be published to a factory instance to have any effect.

Pipeline annotation: A label, added to a pipeline, that appears in the log of subsequent pipeline runs and can be used to filter or group log data. Multiple annotations can be added to a pipeline.

• **Trigger annotation**: A label, added to a trigger, providing functionality analogous to a pipeline annotation.

• **Activity user property**: A name-value pair, added to a pipeline activity, that appears in the log of subsequent pipeline runs. Multiple user properties can be added to an activity. The Copy data activity supports two auto-generated properties that identify its runtime source and sink.

• **Azure Monitor**: Monitoring service used to collect, analyze, and respond to data from Azure resources.

• **Metric**: Automatically maintained count of a given system property over a period of time, emitted to and logged by Azure Monitor.

• **Log Analytics**: Azure Monitor component that enables sophisticated analysis of system logs and metrics.

• **Log Analytics workspace**: Identified Log Analytics provision, to which Azure resource logs and metrics can be sent for analysis and longer-term storage.

• **Diagnostic setting**: Per-resource configuration information identifying log data and metrics to be sent to other storage services, for example, a Log Analytics workspace.

• **Kusto**: Query language used to interrogate data stored in Log Analytics and Azure Data Explorer.

• **Tabular expression statement**: Kusto query expression that returns a result set. Every Kusto query must contain a tabular expression statement.

• **Log Analytics workbook**: A notebook-like interface for querying Log Analytics data, allowing code and text cells to be interleaved to create narrative reports.

• **Azure Data Explorer**: Analytics service for near real-time, large-scale analysis of raw data. Like Log Analytics, the service accepts read-only queries written in Kusto.

• **Alerts**: Azure Monitor supports the raising of alerts in response to configured metric or custom query output thresholds.

• **Alert rule**: Information that defines an alert – its scope (what to monitor), its conditions (when an alert should be raised), and its actions (who to notify and/or what to do when an alert is raised).

• **Signal**: Measure used to evaluate an alert condition.

• **Action group**: Defines a collection of notifications and actions, used to specify the action component of an alert rule.