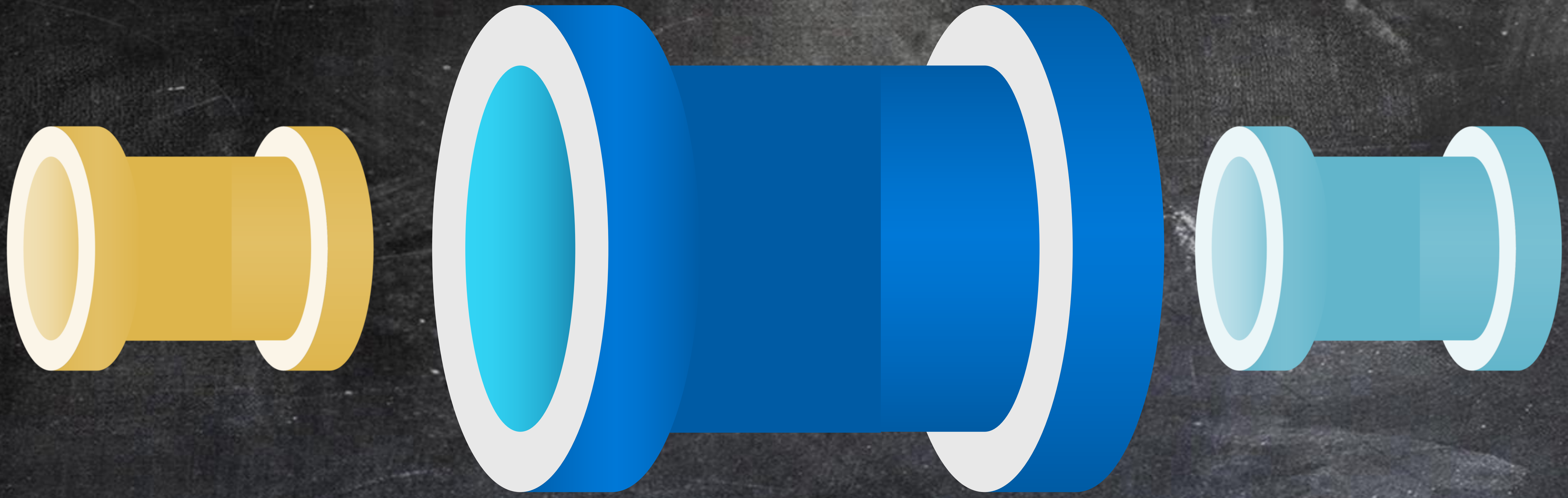
Cloud Formations

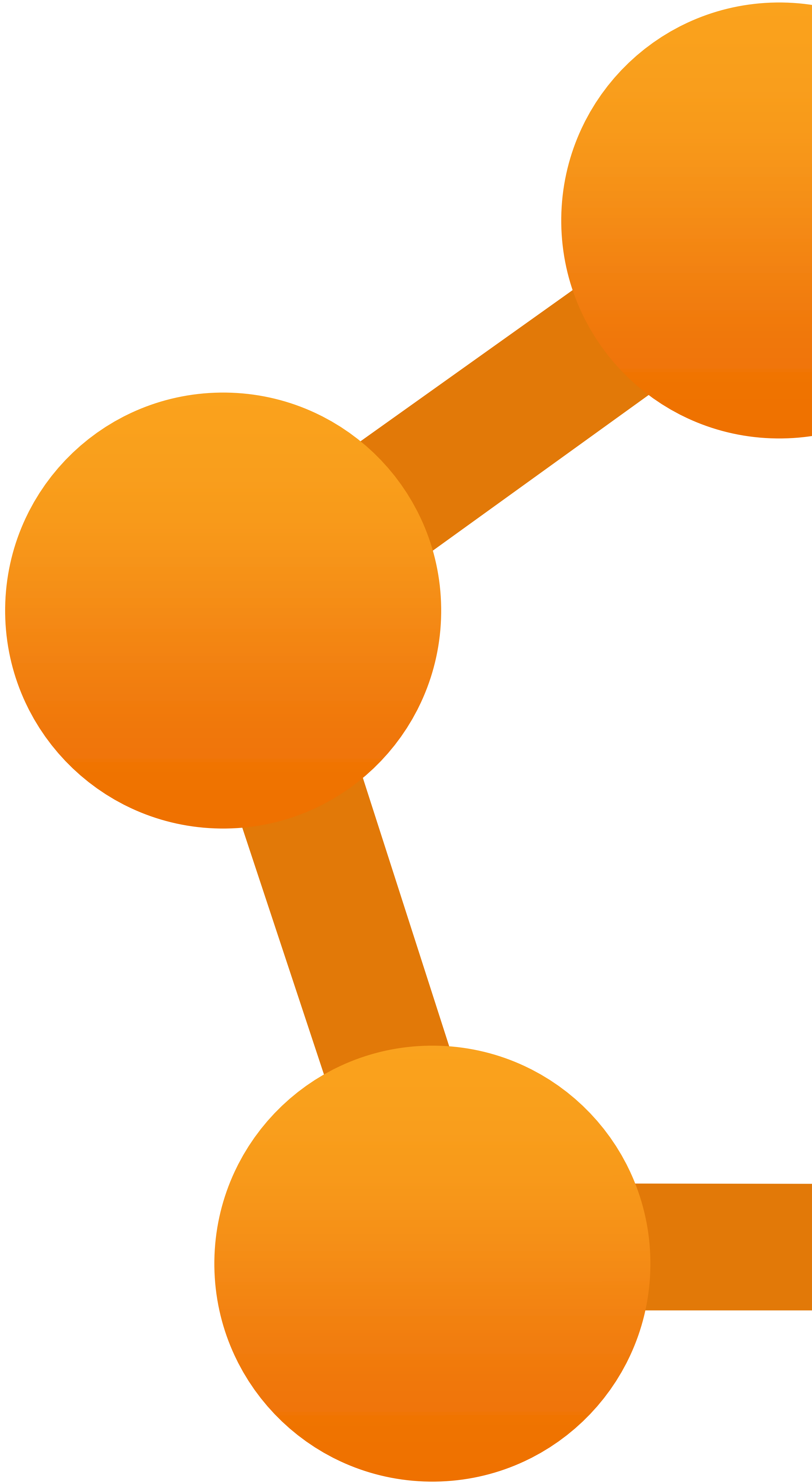
Proudly presents...

Integration Pipelines



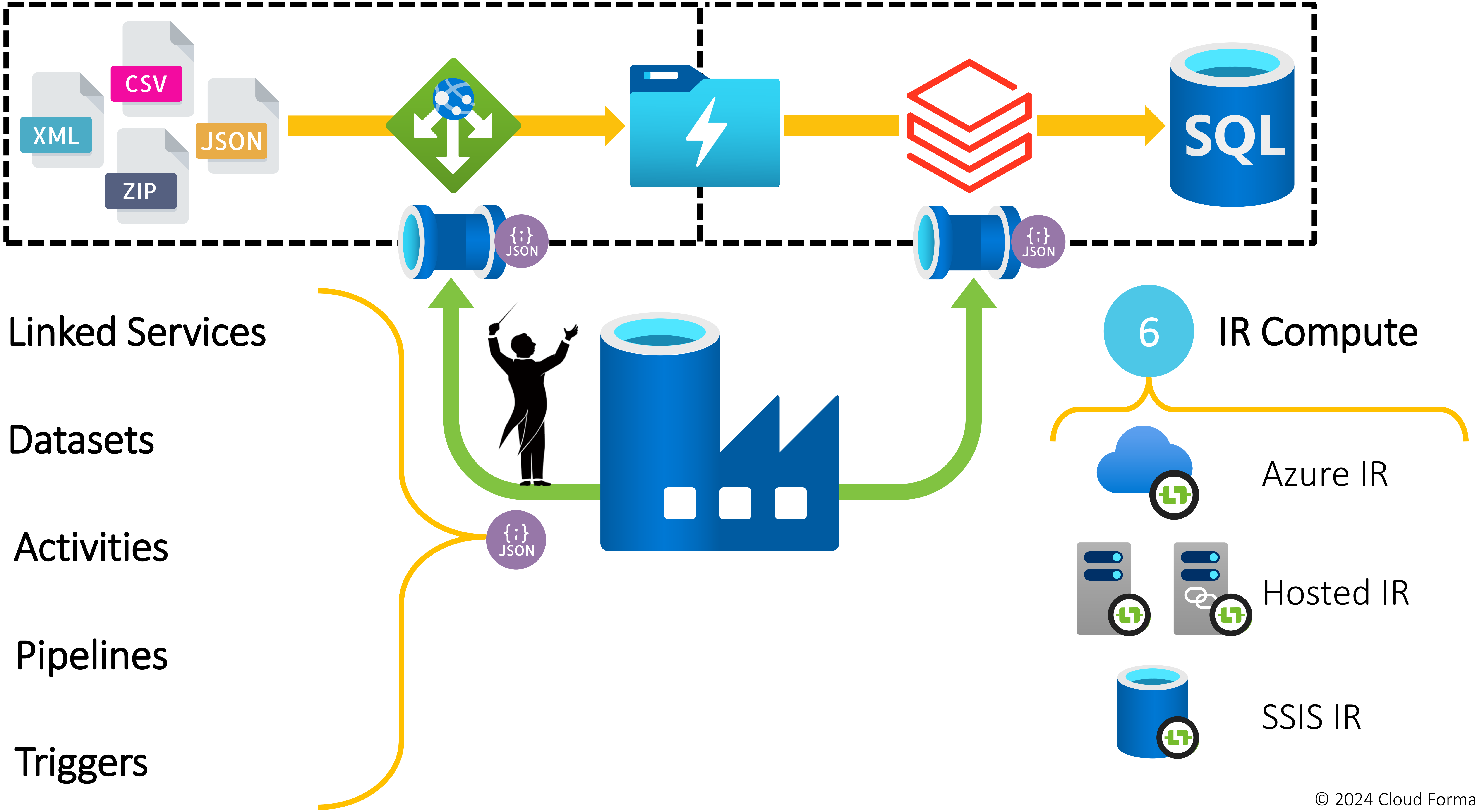
Module 6 – Execution Parallelism

Control Flow Scale Out



Cloud Formations

Data Factory Core Components



1 Linked Services

2 Datasets

3 Activities

4 Pipelines

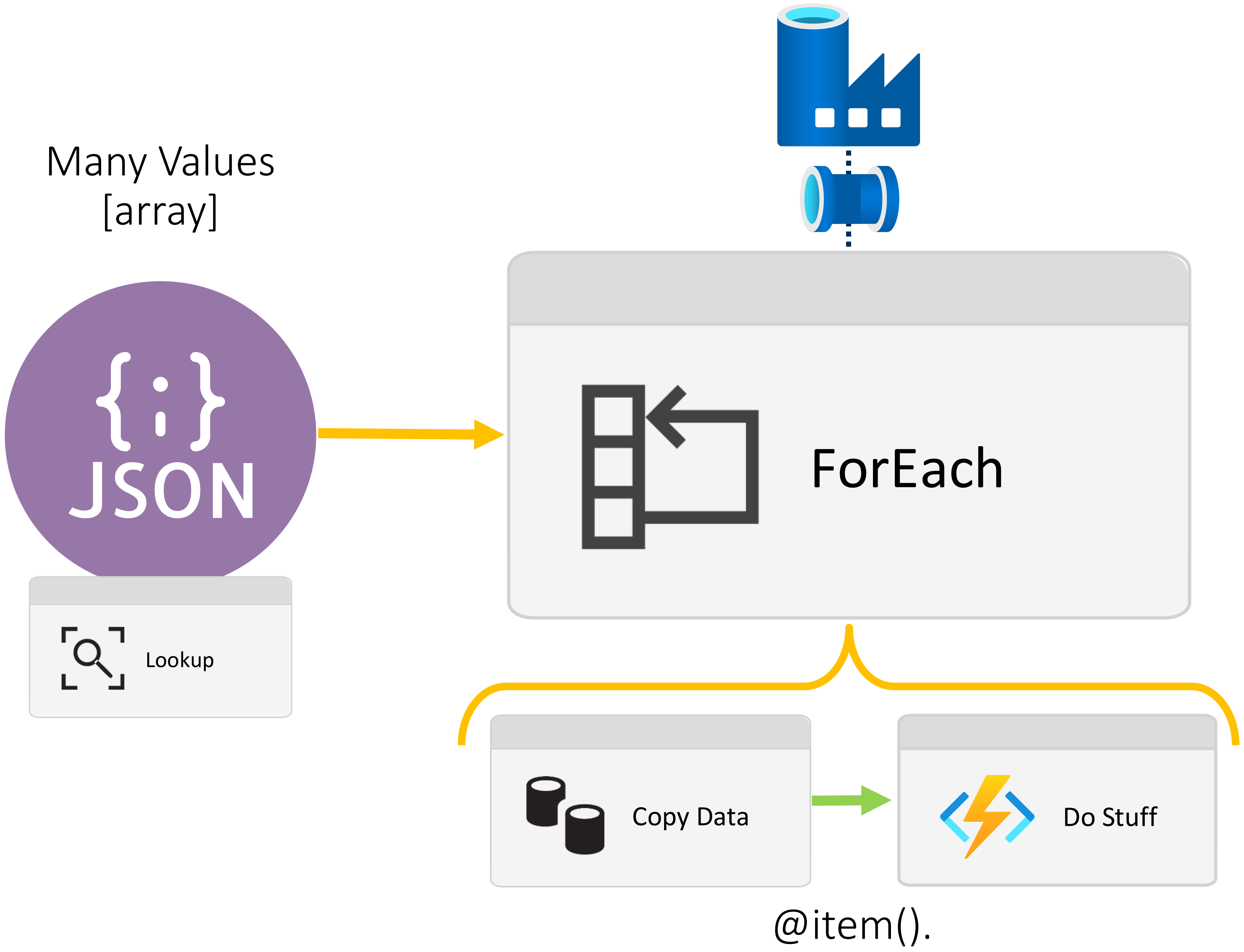
5 Triggers

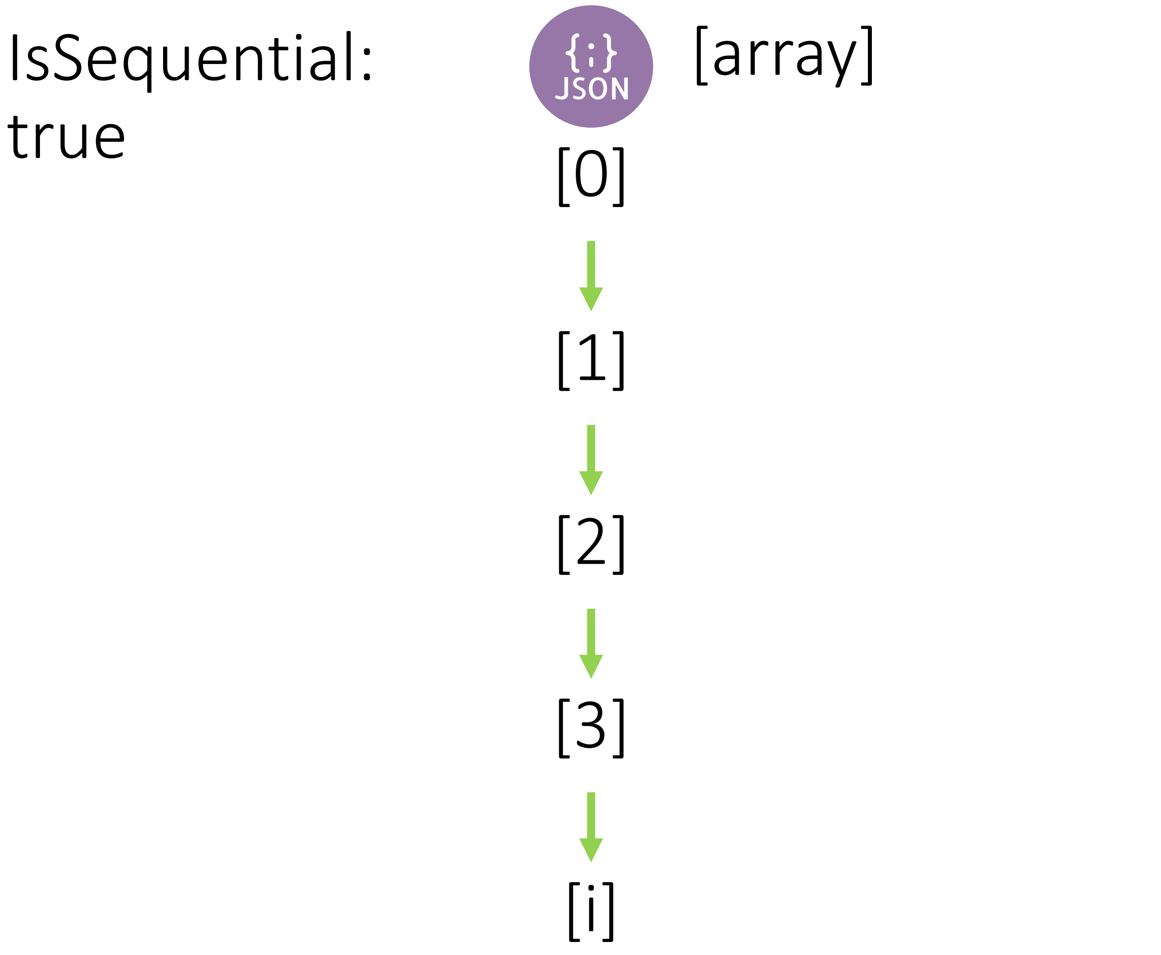
6 IR Compute

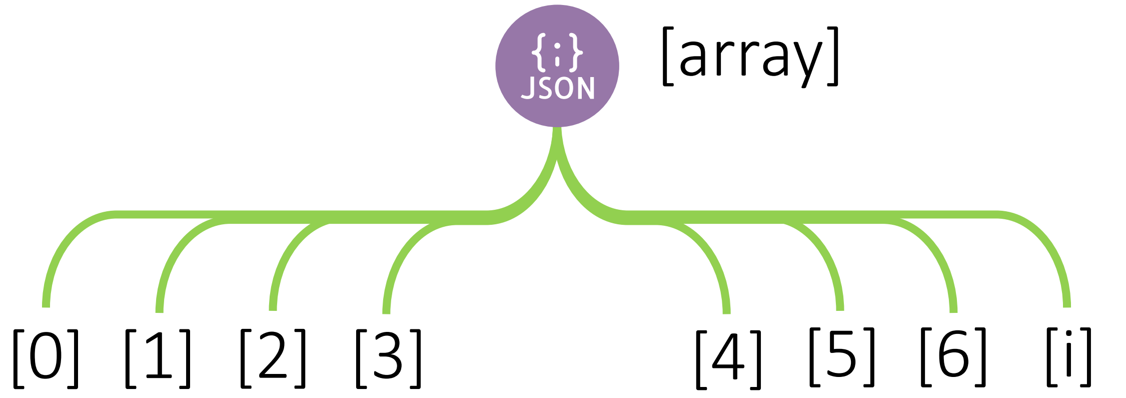
- Azure IR
- Hosted IR
- SSIS IR

For Each

Iterating over other control flow activities







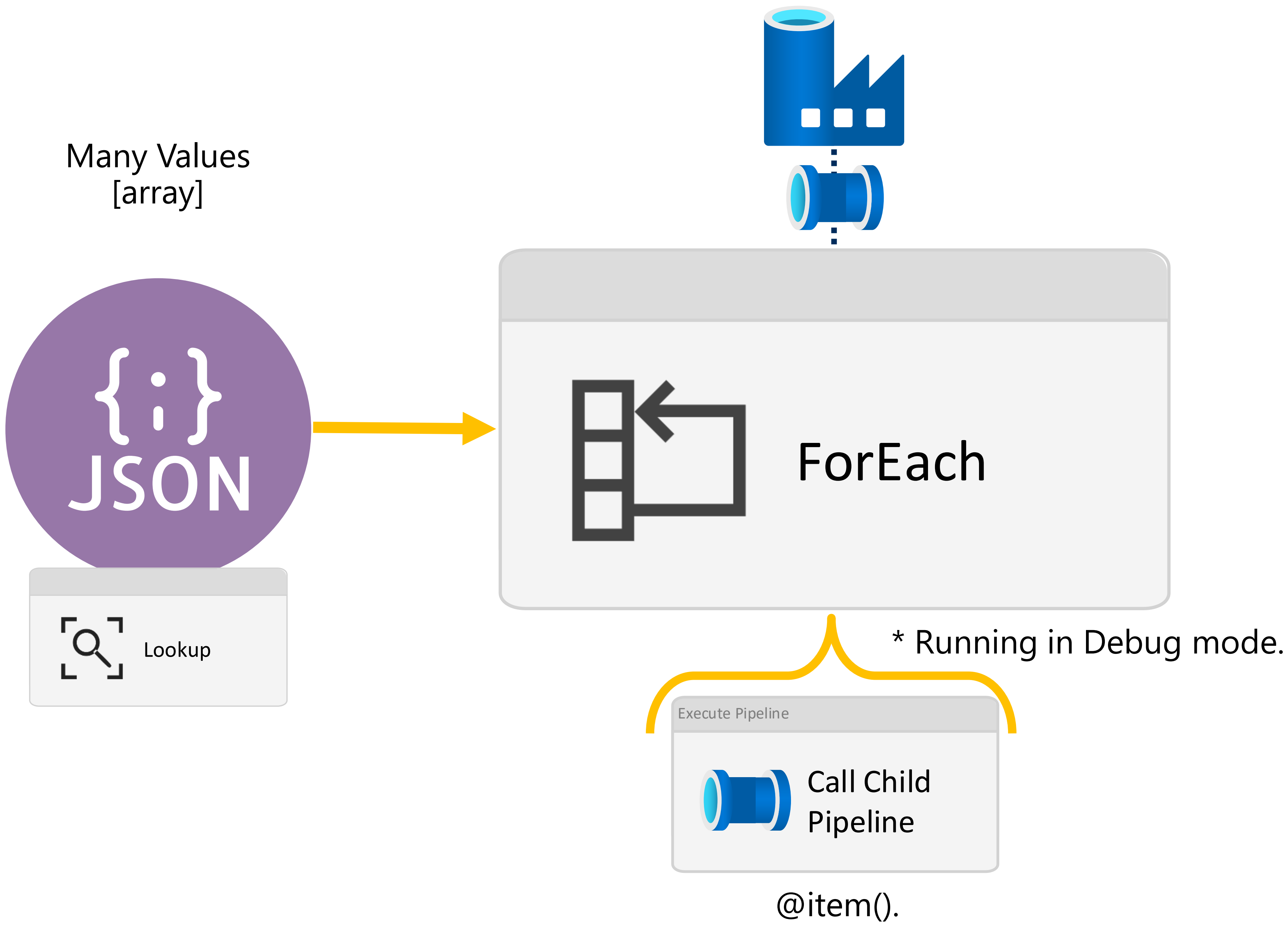
Batch Count Default: 20

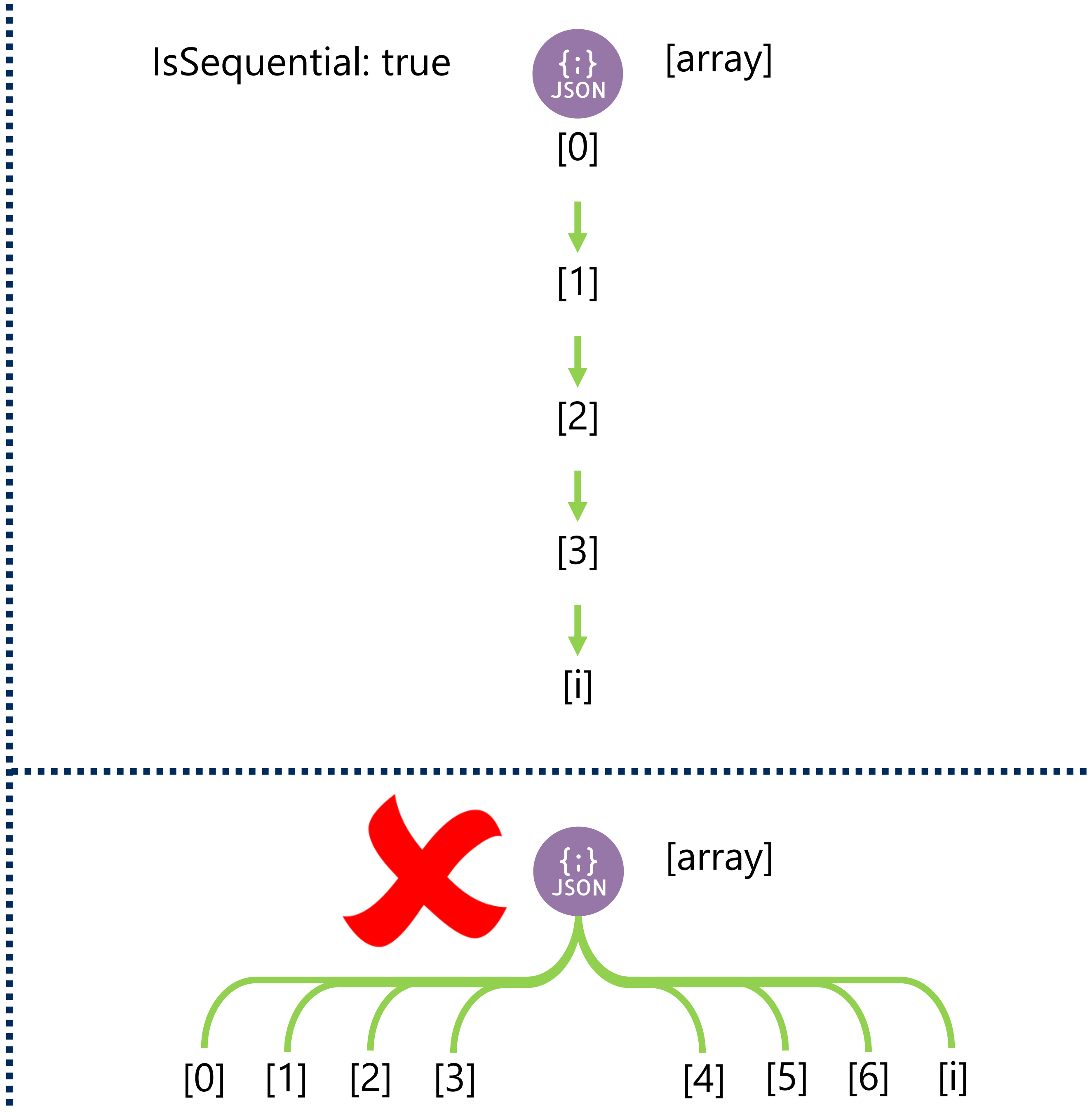
Batch Count Max: 50

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-for-each-activity

For Each Activity

Iterating over other control flow activities



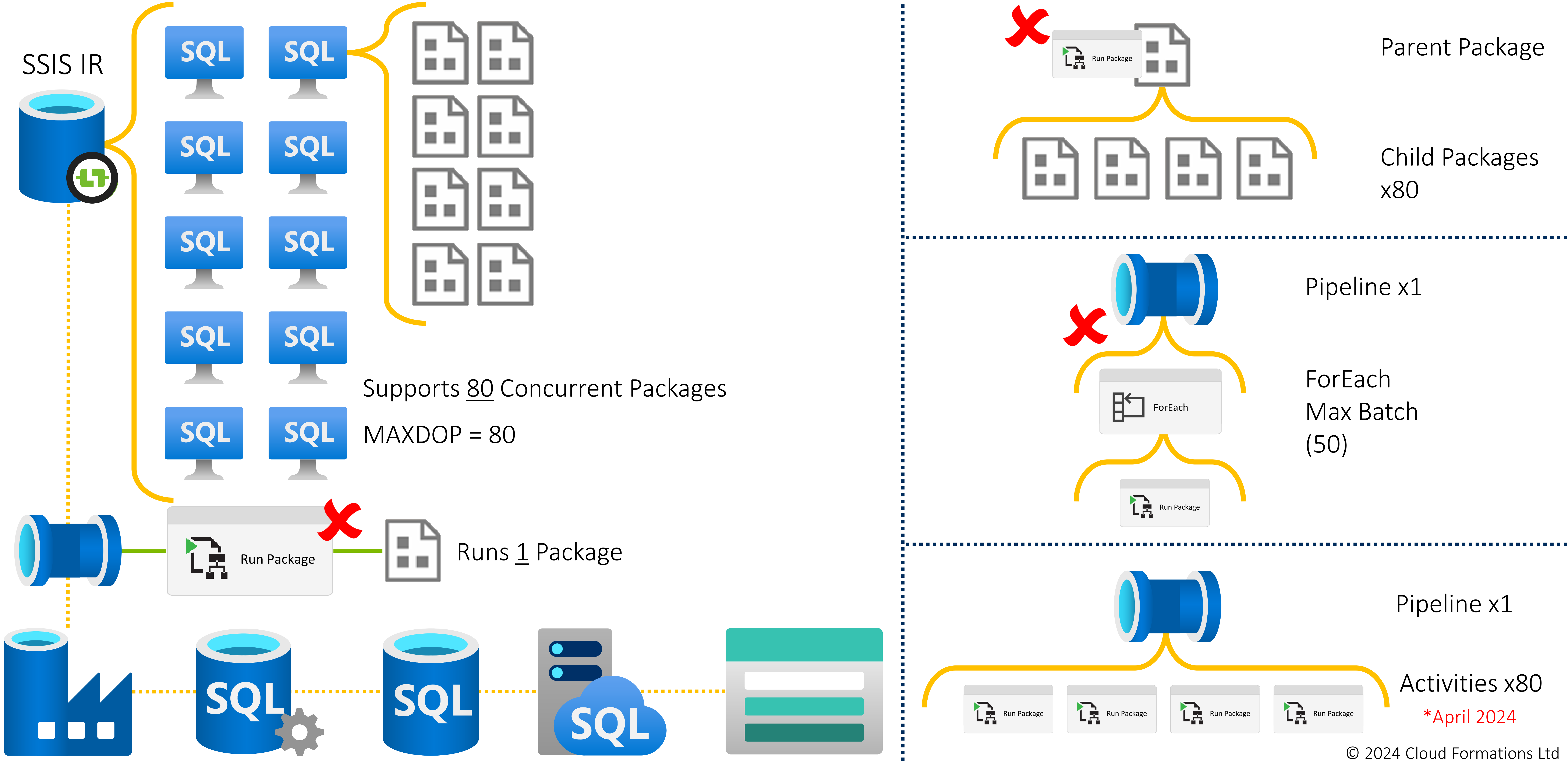


Batch Count Default: 20

Batch Count Max: 50

Cloud Formations - Knowledge Transfer & Training

Problem: Using All Of The SSIS IR Compute



SSIS IR

Supports 80 Concurrent Packages

MAXDOP = 80

Runs 1 Package

Parent Package

Child Packages x80

Pipeline x1

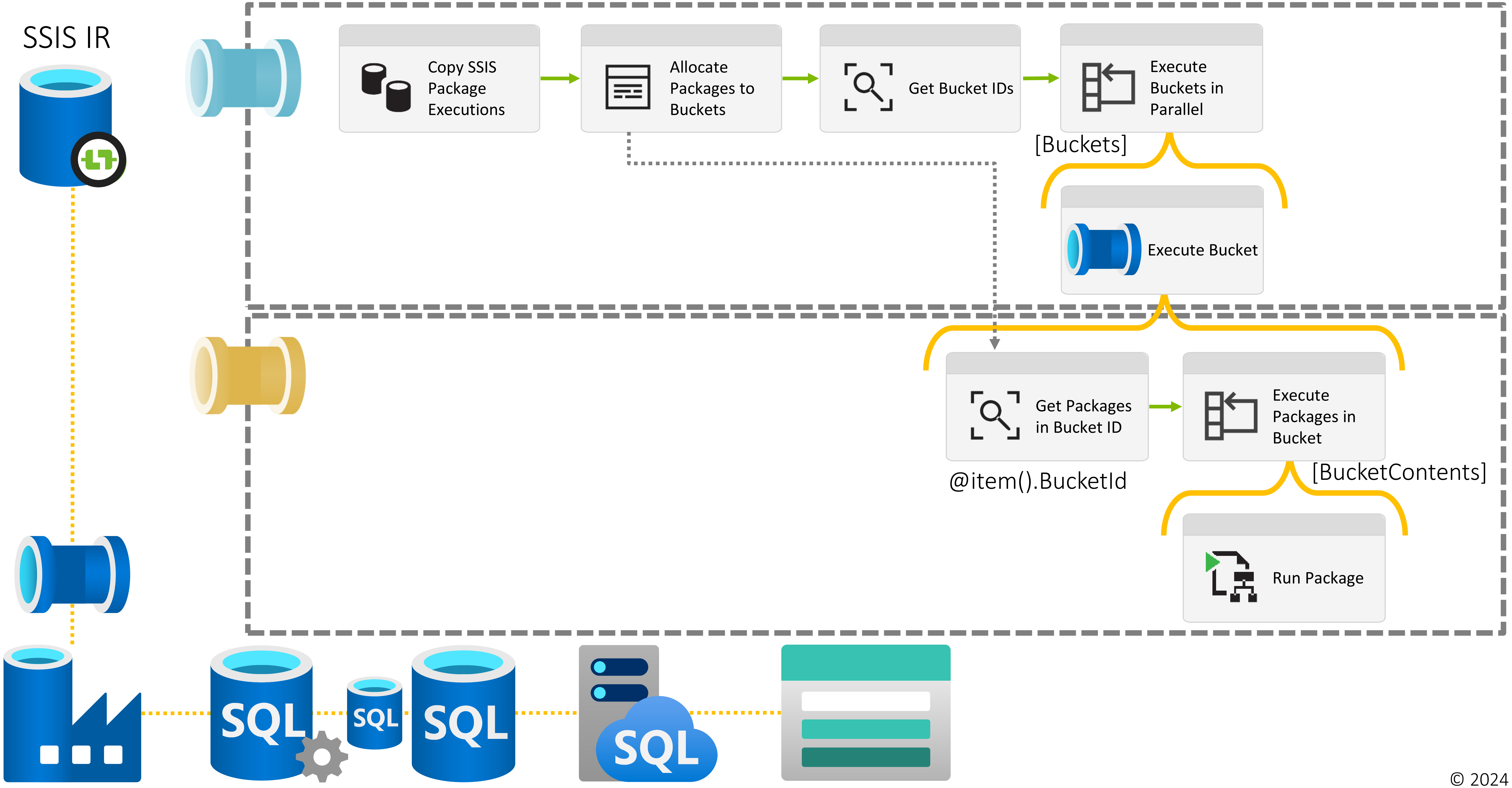
ForEach Max Batch (50)

Pipeline x1

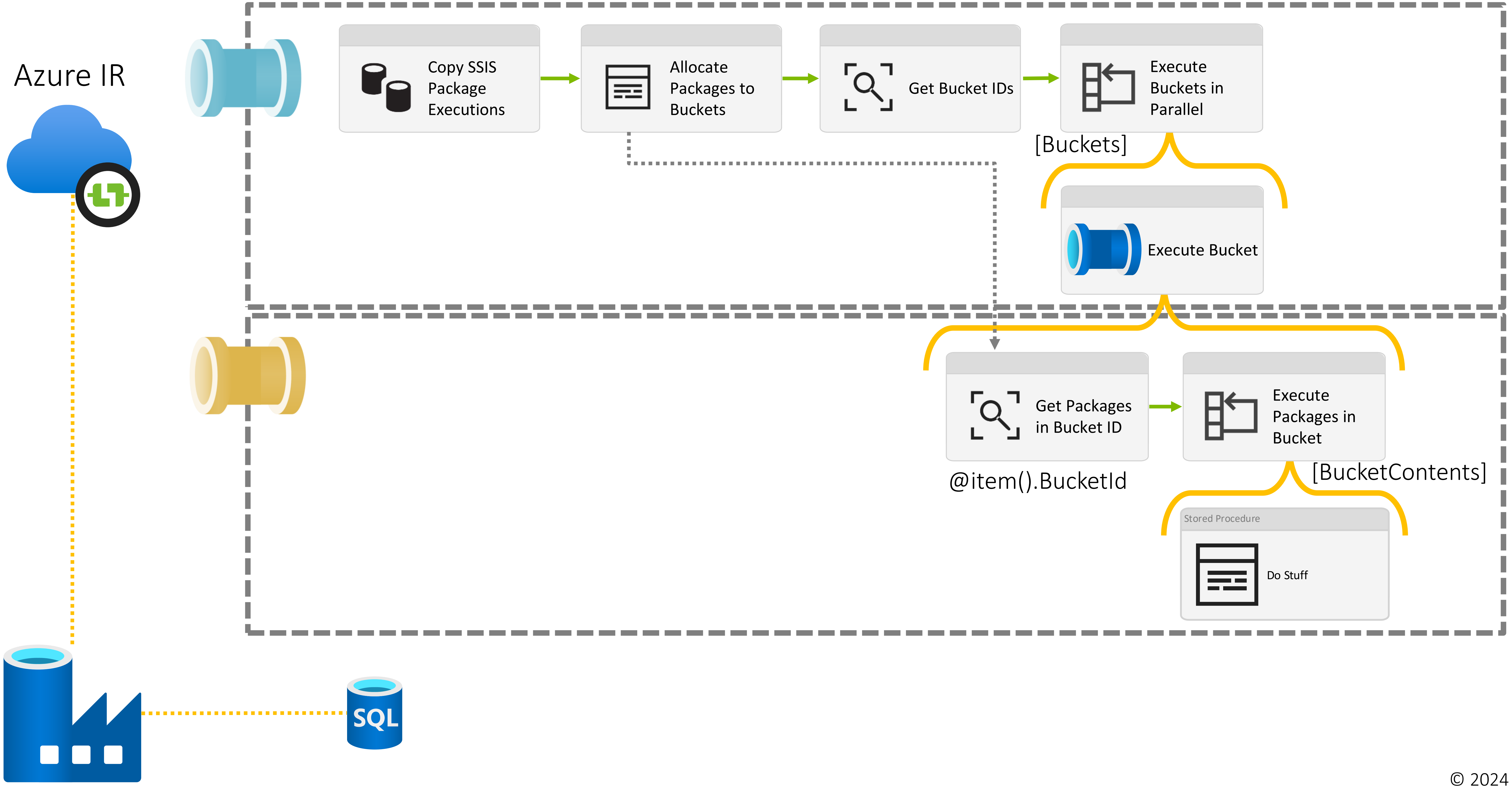
Activities x80

*April 2024

Nested ForEach Activities & Bucket Metadata


SSIS IR
Copy SSIS Package Executions
Allocate Packages to Buckets
Get Bucket IDs
Execute Buckets in Parallel
[Buckets]
Execute Bucket
Get Packages in Bucket ID
Execute Packages in Bucket
@item().BucketId
[BucketContents]
Run Package
SQL
SQL
SQL
SQL

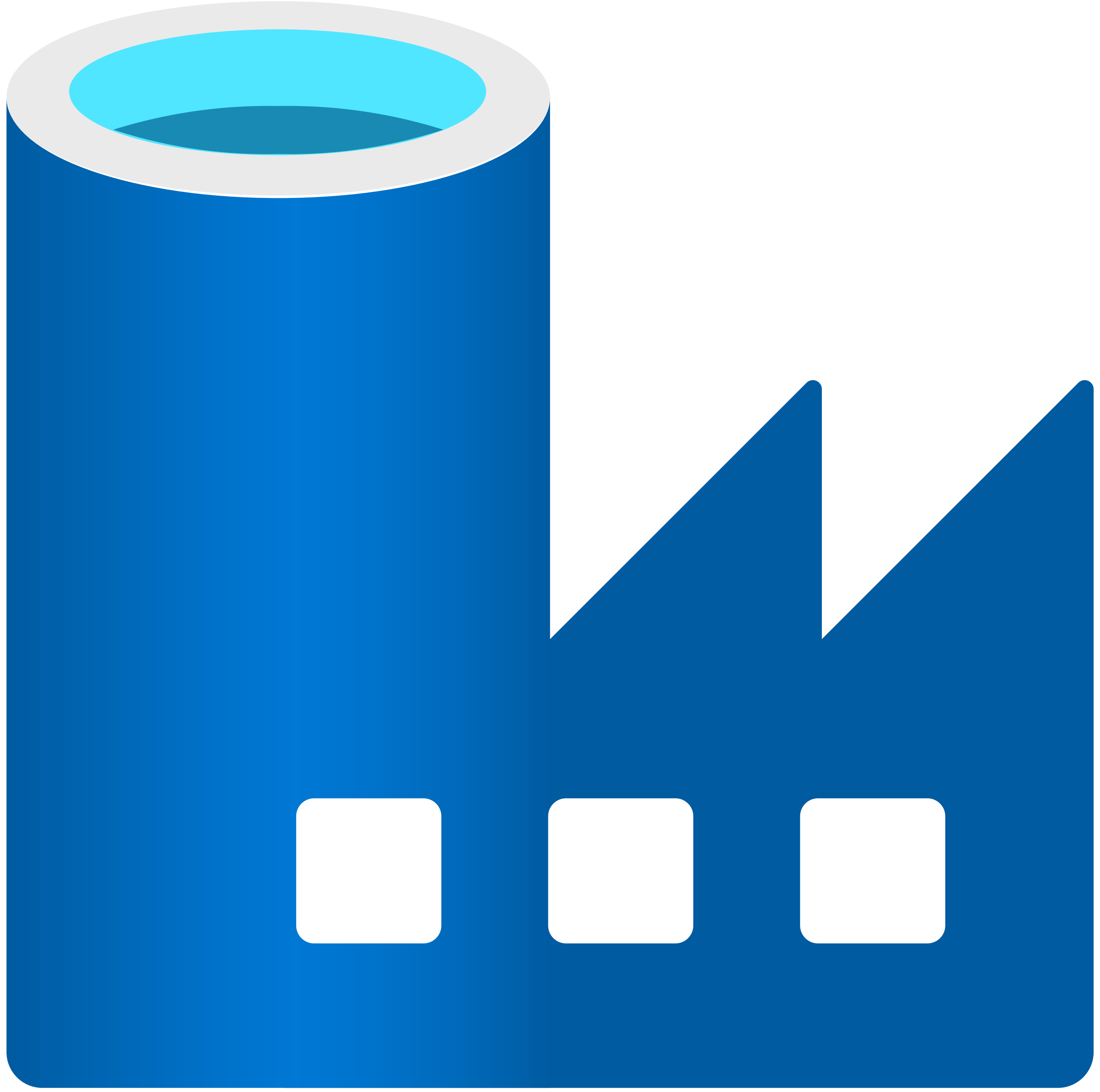
A General Pattern for Scaling Out



Azure IR
Copy SSIS Package Executions
Allocate Packages to Buckets
Get Bucket IDs
Execute Buckets in Parallel
[Buckets]
Execute Bucket
Get Packages in Bucket ID
Execute Packages in Bucket
@item().BucketId
[BucketContents]
Stored Procedure
Do Stuff
SQL

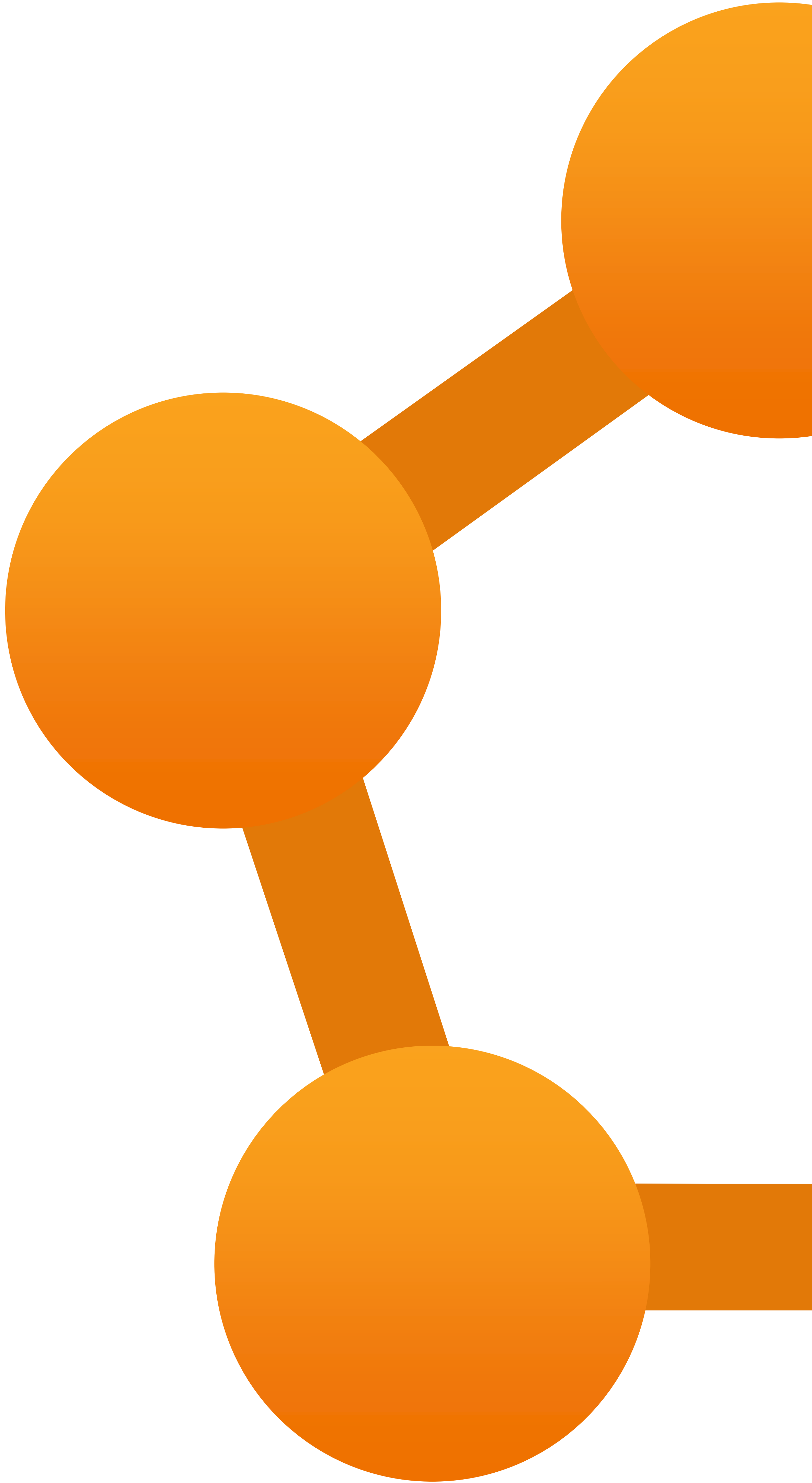
Cloud Formations - Knowledge Transfer & Training

Azure Portal



Module 6 – Execution Parallelism

Concurrency Limits vs Internal vs External Activities



Cloud Formations

Data Factory Limitations

Resource Limits

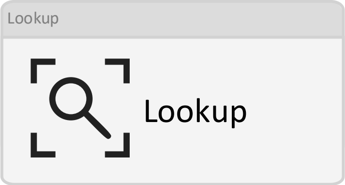
mrpaulandrew.com/2020/01/29/azure-data-factory-resource-limitations/

https://github.com/MicrosoftDocs/azure-docs/blob/main/includes/azure-data-factory-limits.md

800 Data Factory Instances per Subscription

80 Activities per Data Factory Pipeline

3 Active Data Flow Debug Sessions per Data Factory



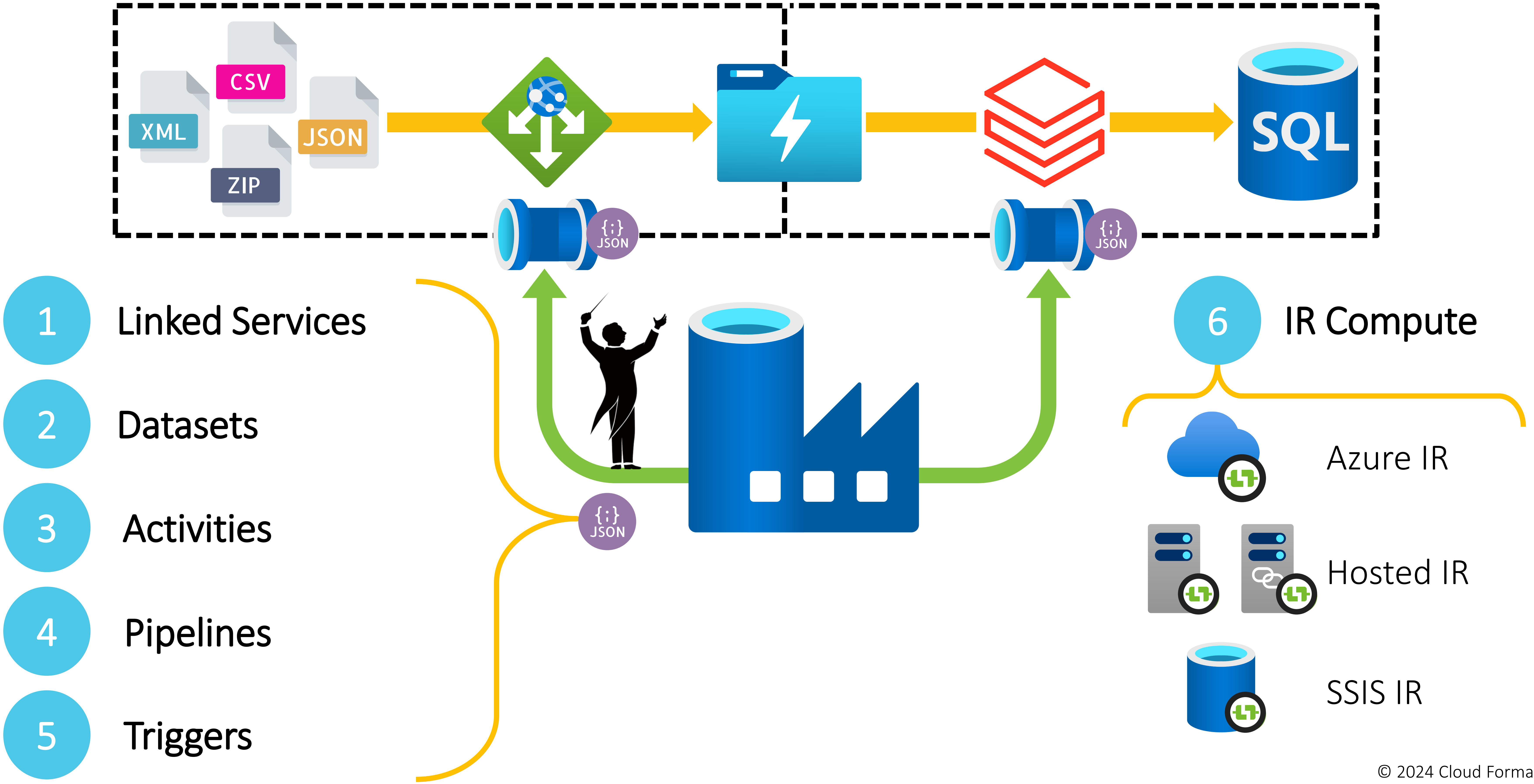
5,000 Rows or 4MB of Data Returned per Lookup (No Error if More)

Minimum Tumbling Window Trigger – 15mins

4min Client Response Timeout Using Azure Functions Activity

5,000 Entities (Components) per Data Factory Instance

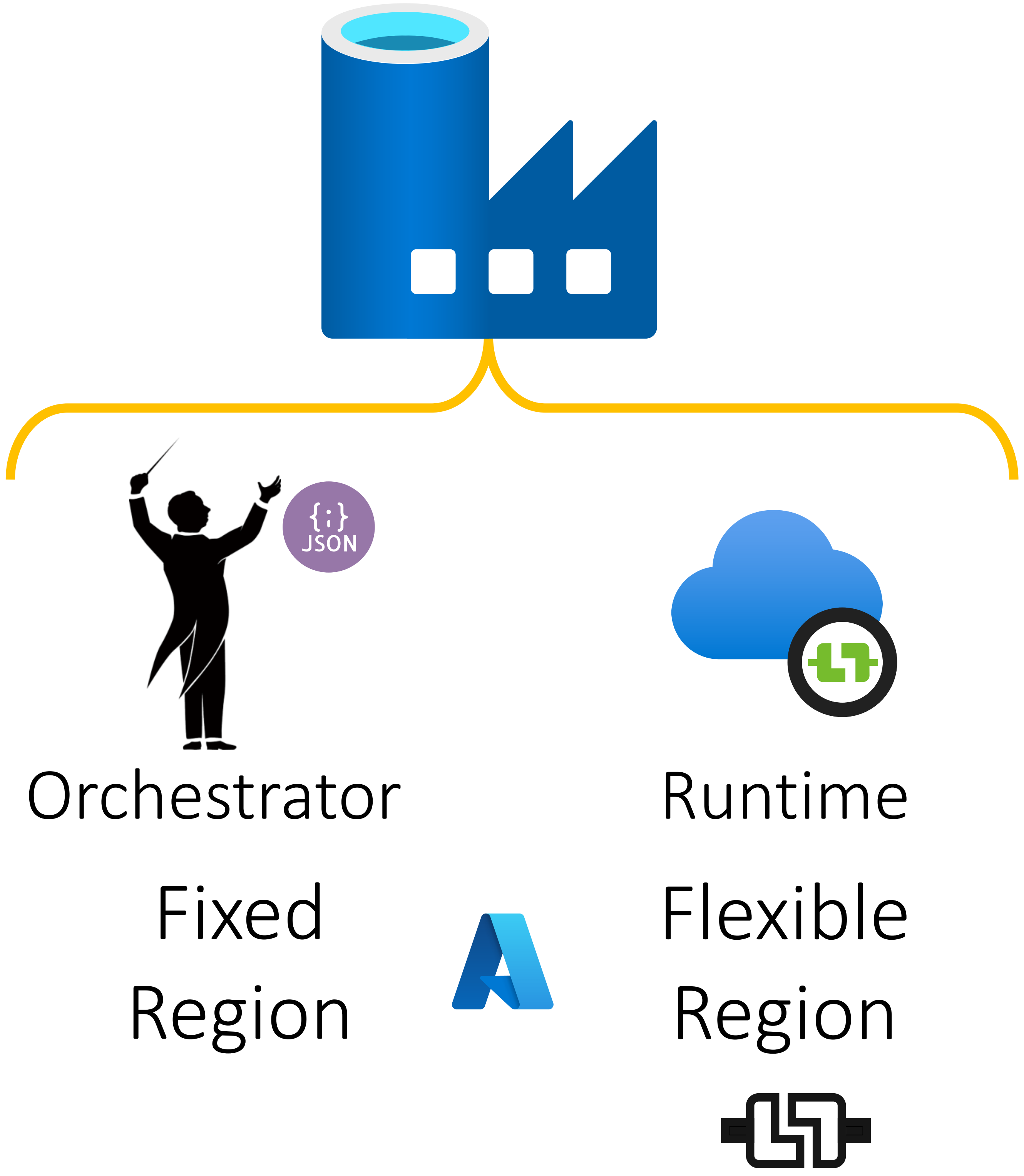
Data Factory Core Components



Data Factory Core Components

Cloud Formations - Knowledge Transfer & Training

Compute Concurrency



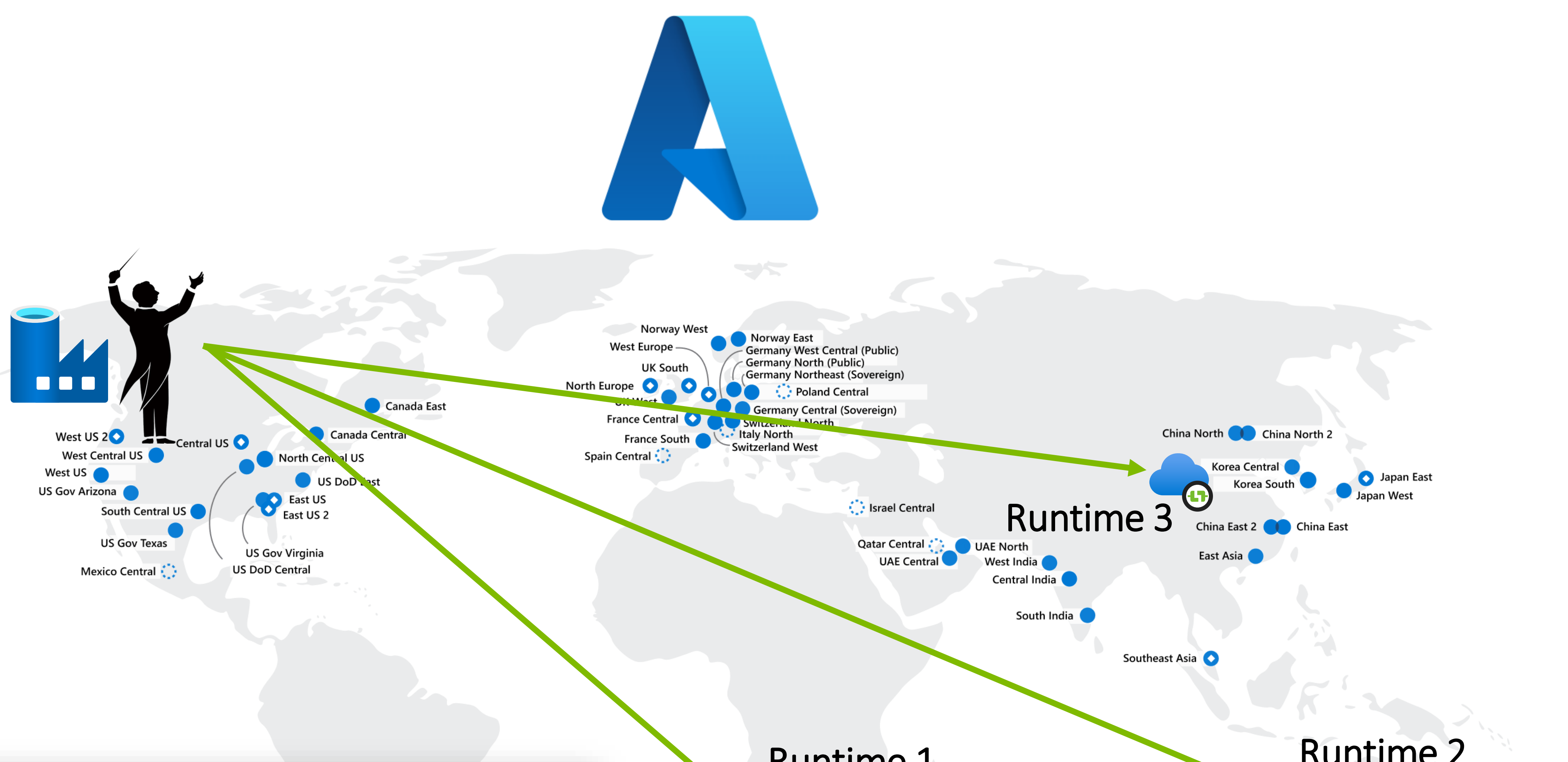
Orchestrator

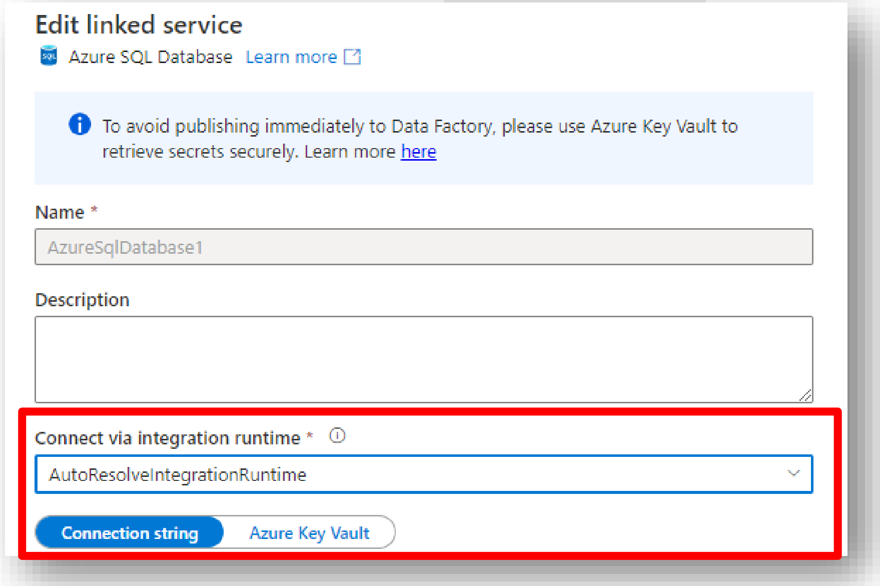
Fixed Region

Runtime

Flexible Region

AutoResolveIntegrationRuntime





Edit linked service

Azure SQL Database Learn more

To avoid publishing immediately to Data Factory, please use Azure Key Vault to retrieve secrets securely. Learn more here

Name *

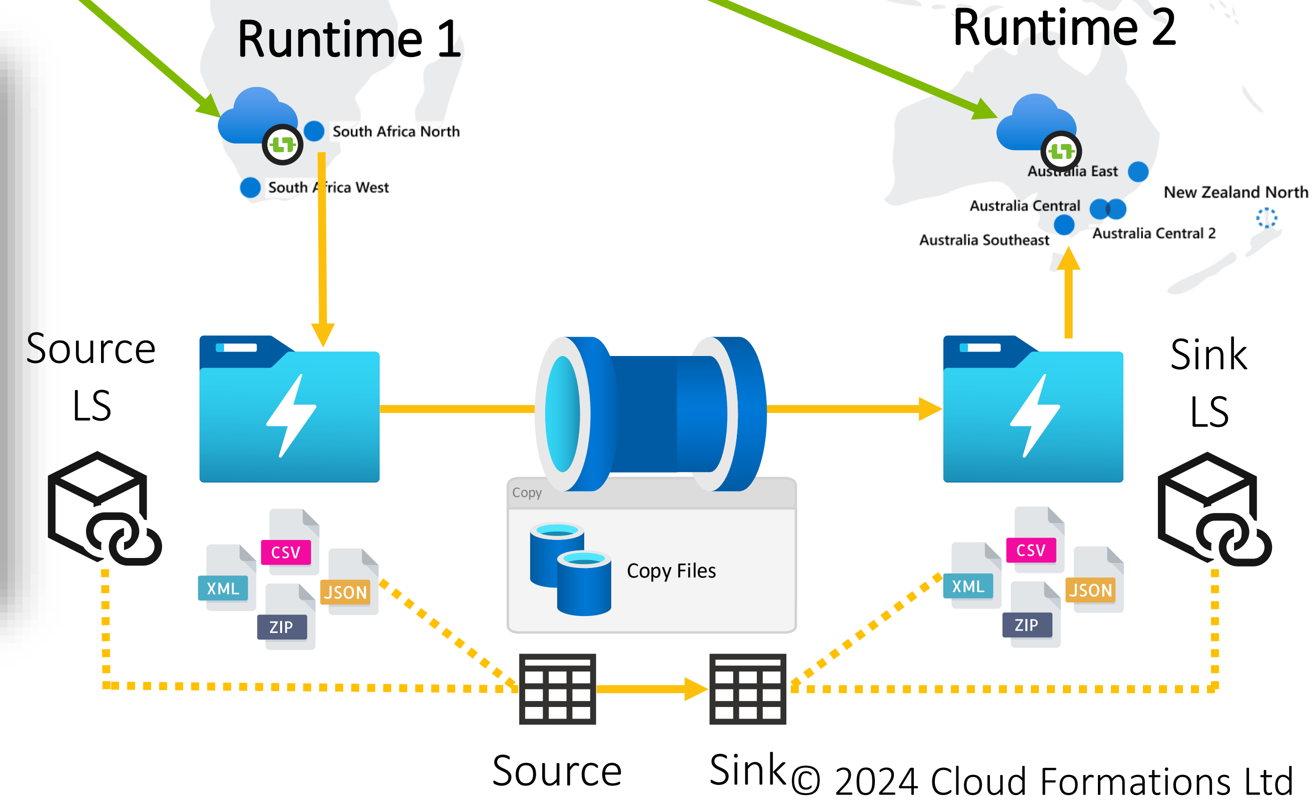
AzureSqlDatabase1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

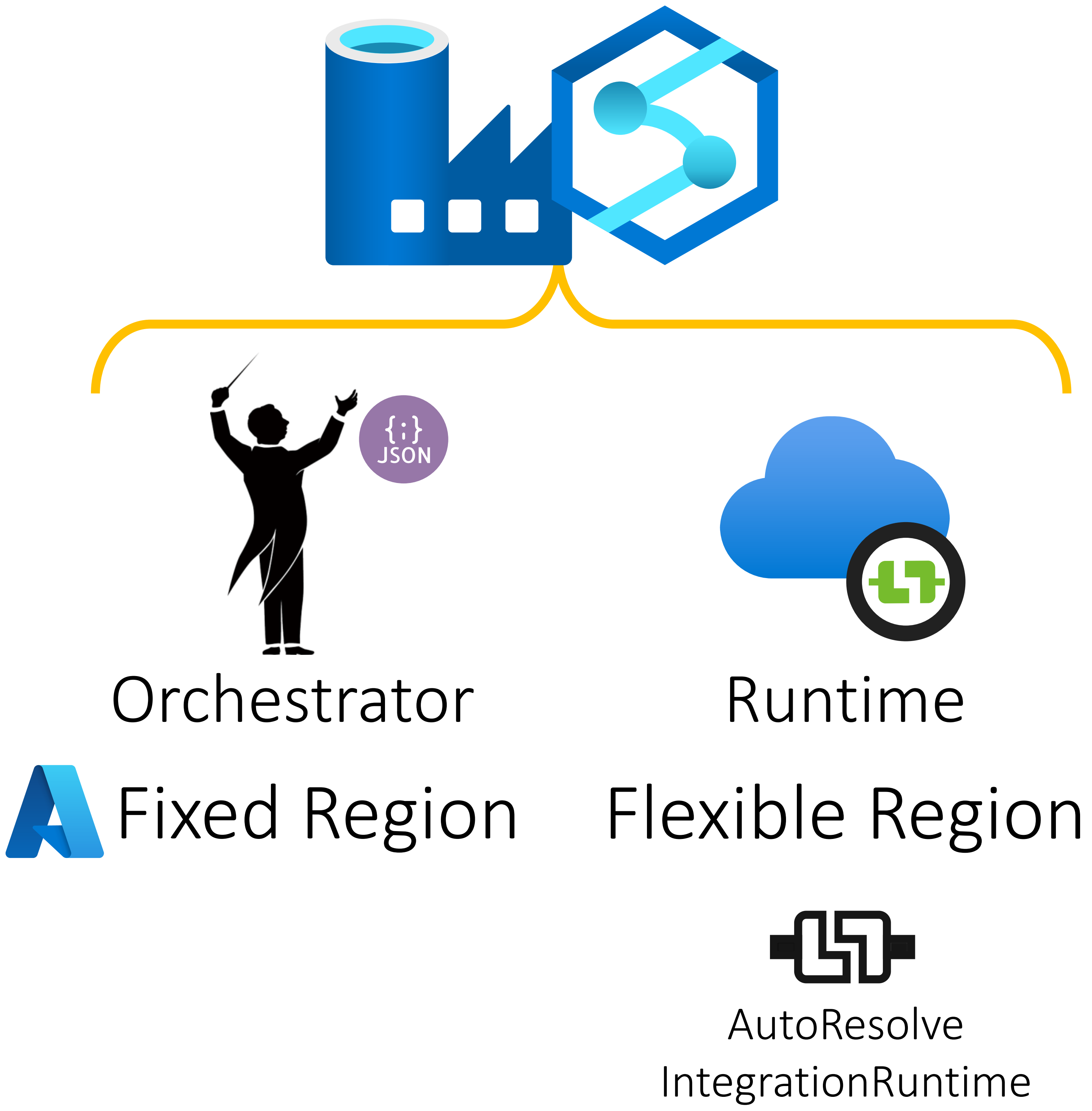
Connection string | Azure Key Vault



Compute Concurrency

Internal vs External Activities

https://mrpaulandrew.com/2020/12/22/pipelines-understanding-internal-vs-external-activities/



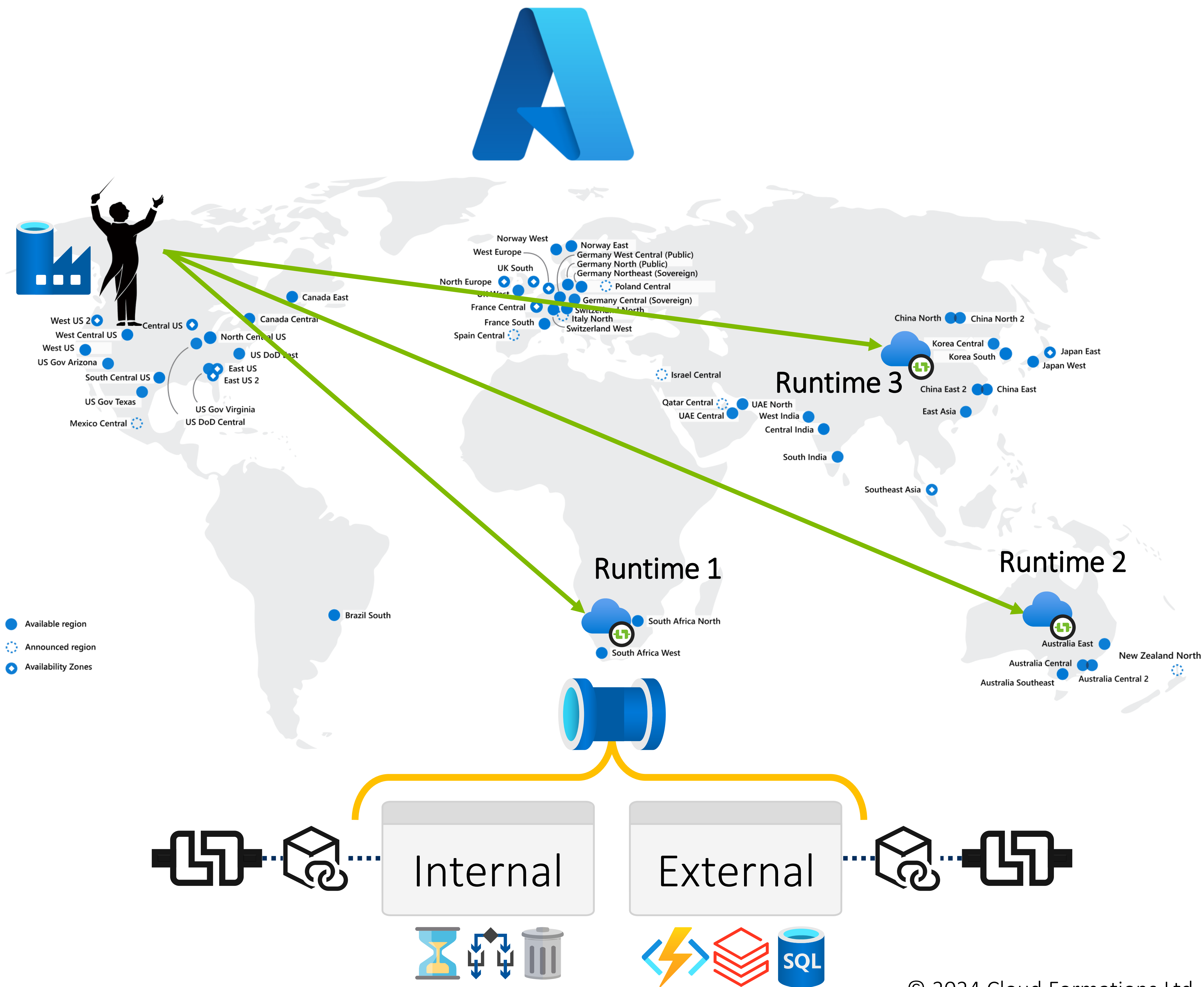
Orchestrator

Runtime

Fixed Region

Flexible Region

AutoResolve IntegrationRuntime



Runtime 3

Runtime 1

Runtime 2

Internal

External

Concurrency – Pipelines vs Activities

Per Subscription, per IR Region

10,000

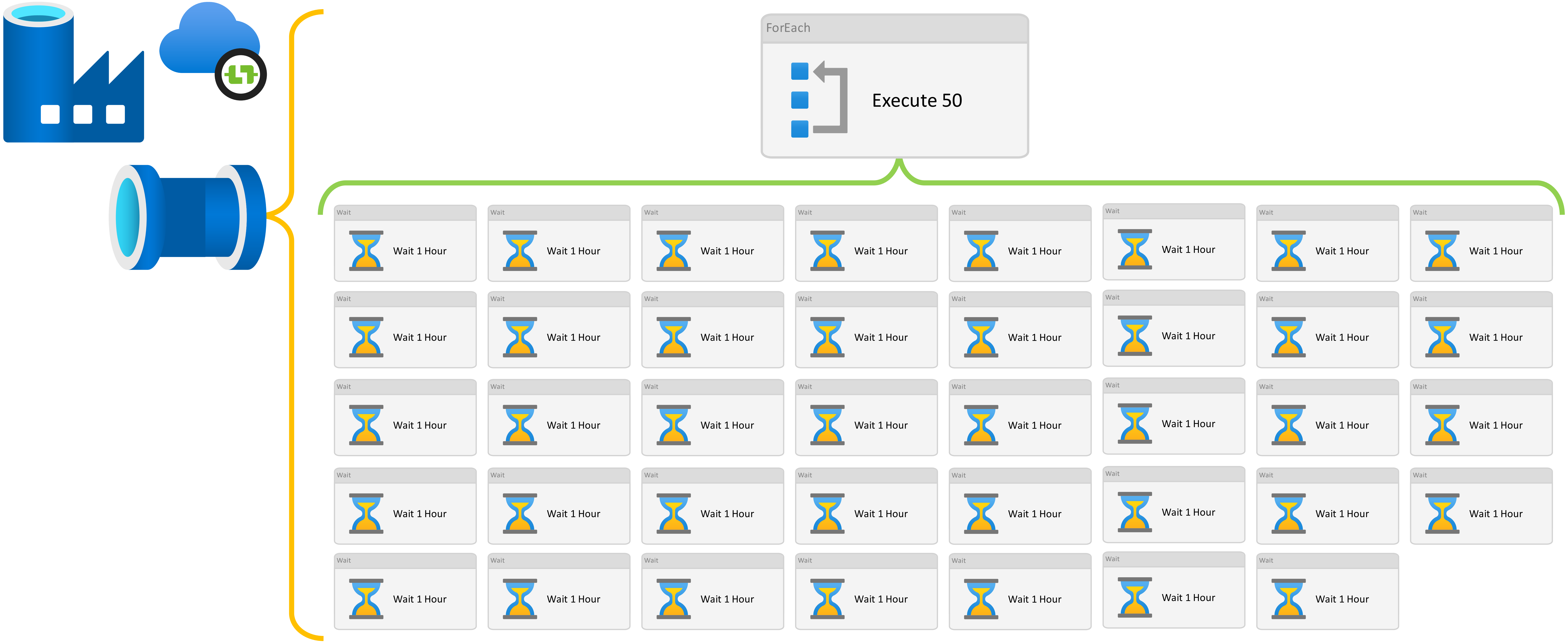
Internal 1,000

External 3,000

https://mrpaulandrew.com/2020/01/29/azure-data-factory-resource-limitations/

Concurrency – Pipelines vs Activities

Example



ForEach

Execute 50

Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour

Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour

Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour

Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour

Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour | Wait 1 Hour

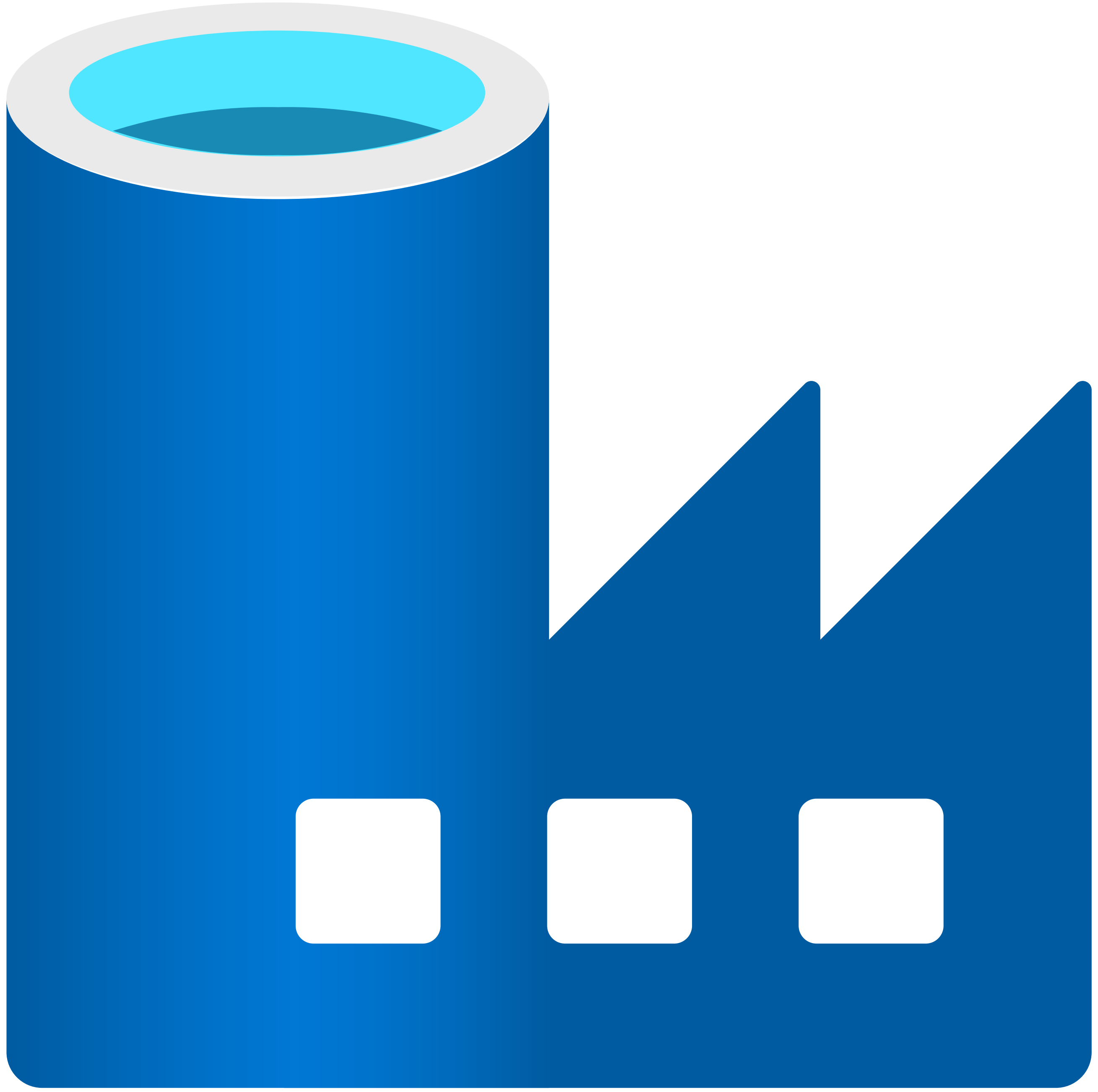
1 IR

1 Pipeline

1 ForEach + (50 Batches X 39 Wait Activities) = 1951 Concurrent Activities

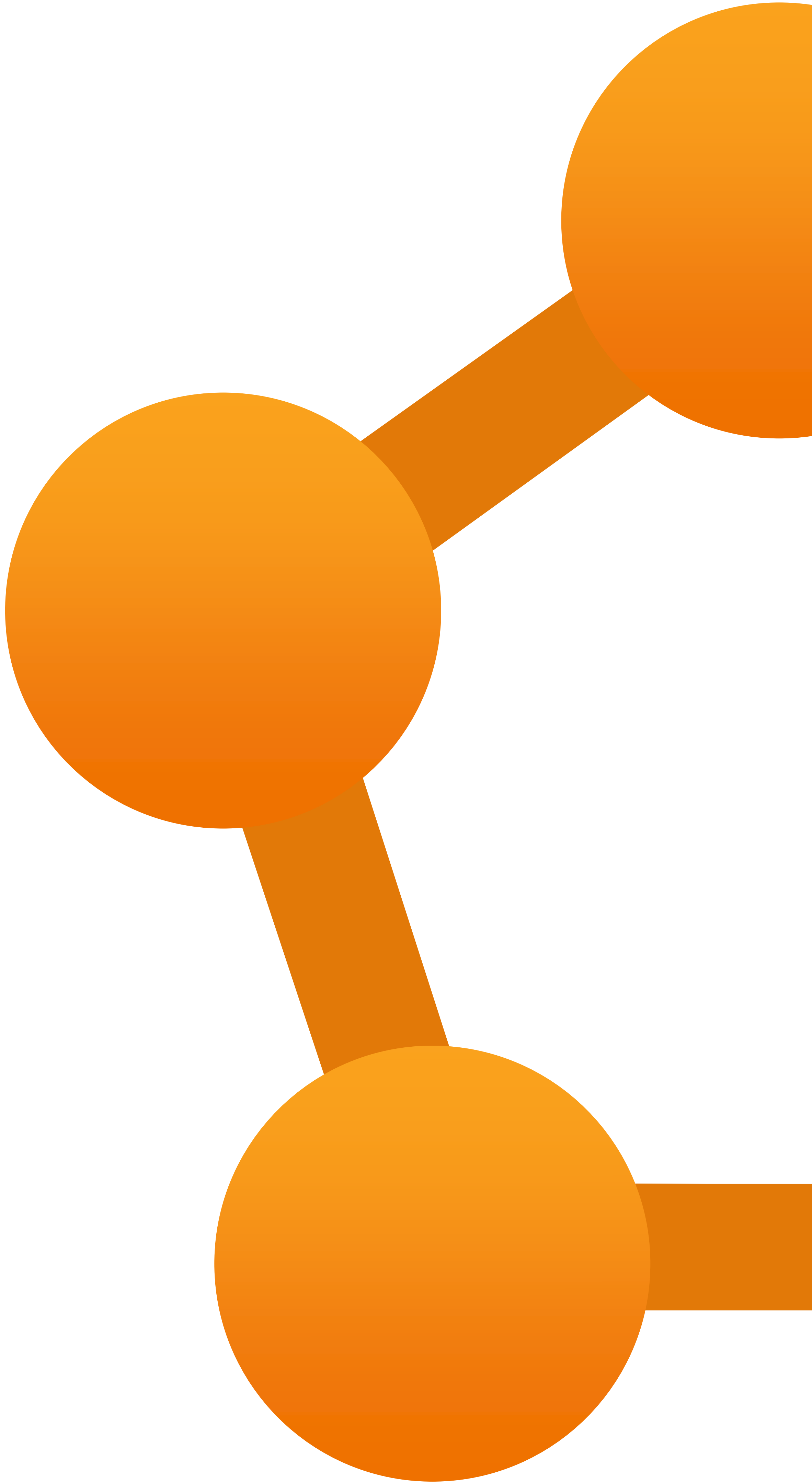
Cloud Formations - Knowledge Transfer & Training

Azure Portal



Module 6 – Execution Parallelism

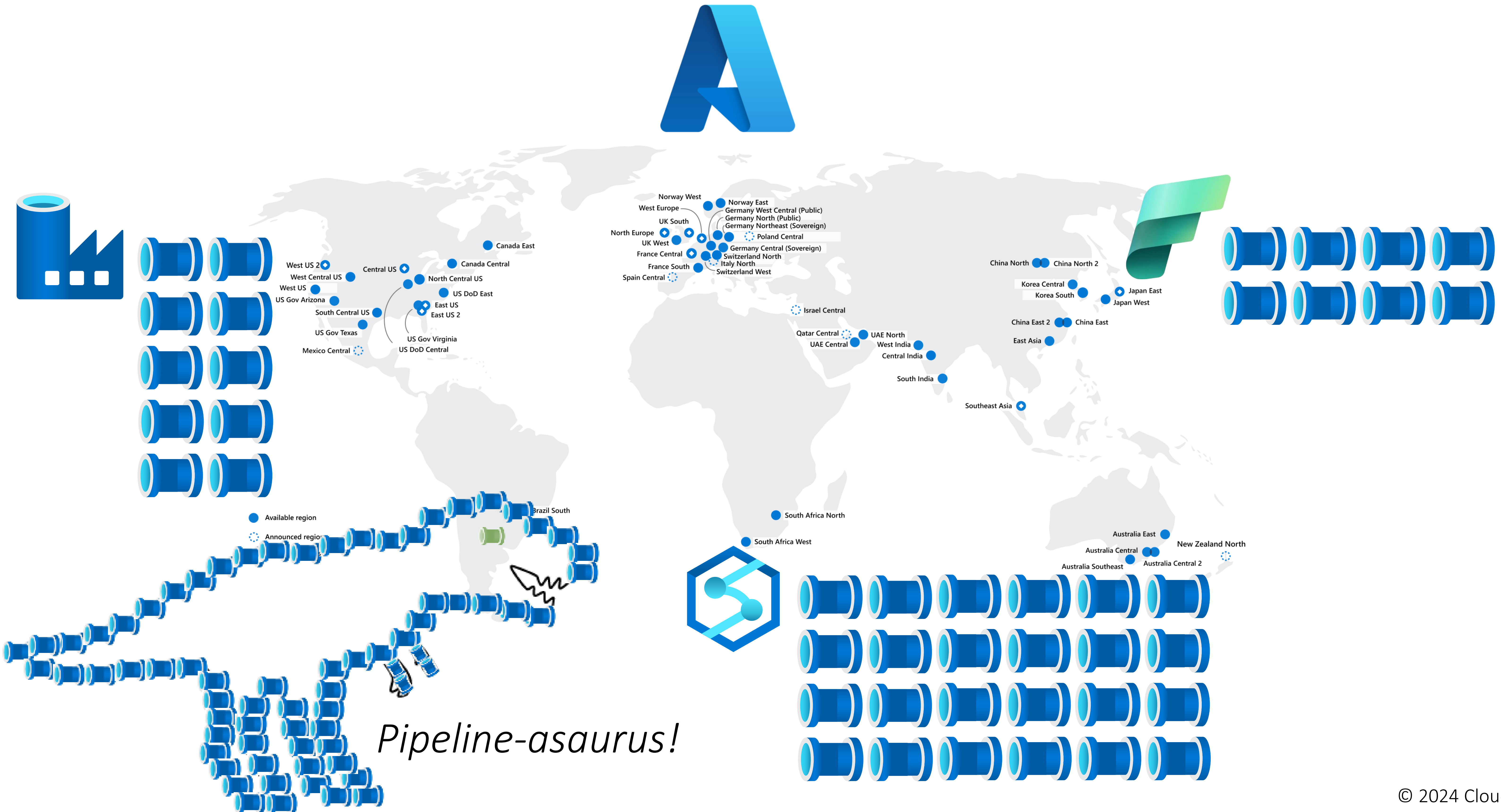
Metadata Driven Frameworks



Cloud Formations

Problem

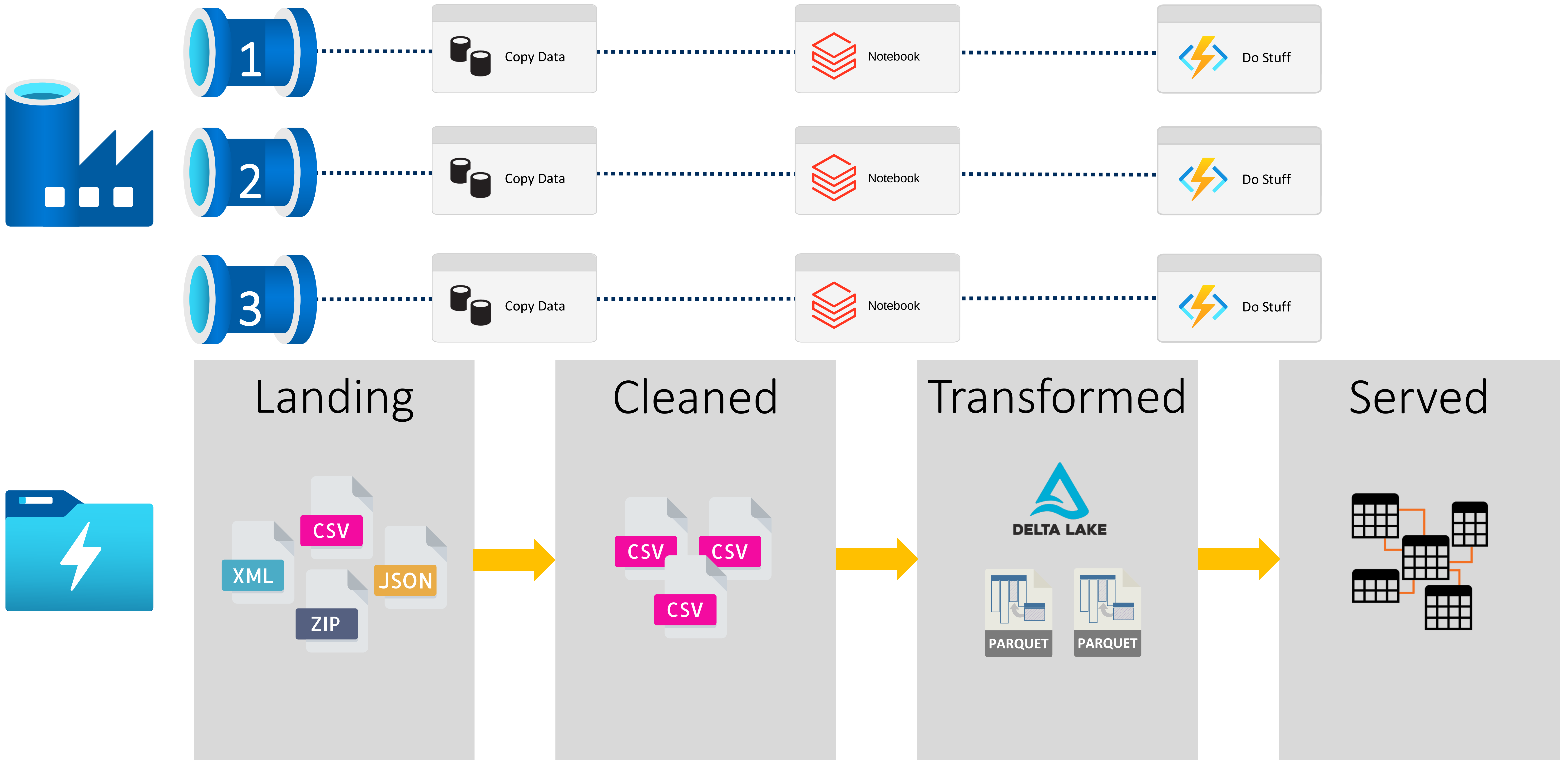
How should we structure and trigger our Integration Pipelines?



Pipeline-asaurus!

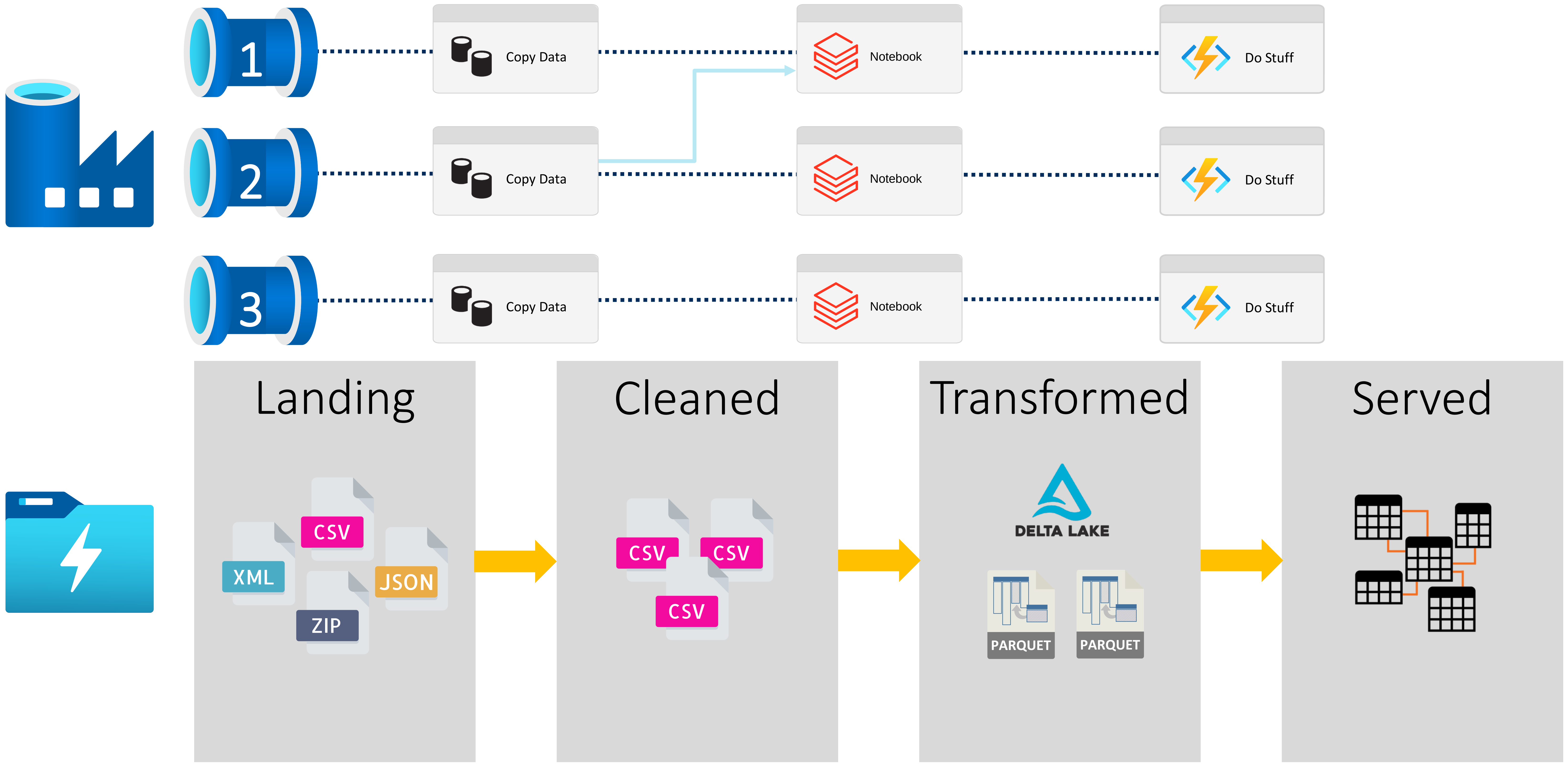
Cloud Formations - Knowledge Transfer & Training

Problem



Cloud Formations - Knowledge Transfer & Training

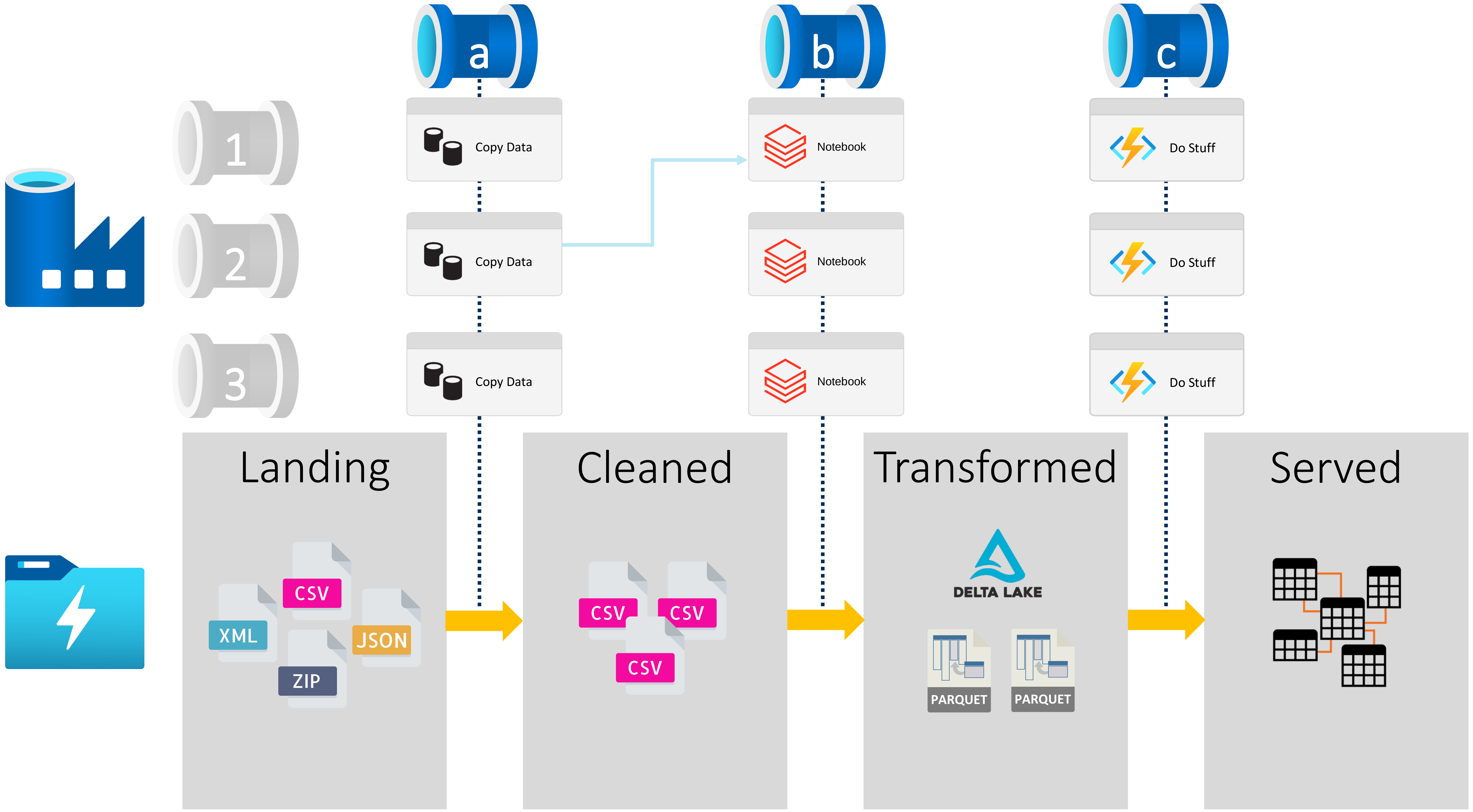
Problem



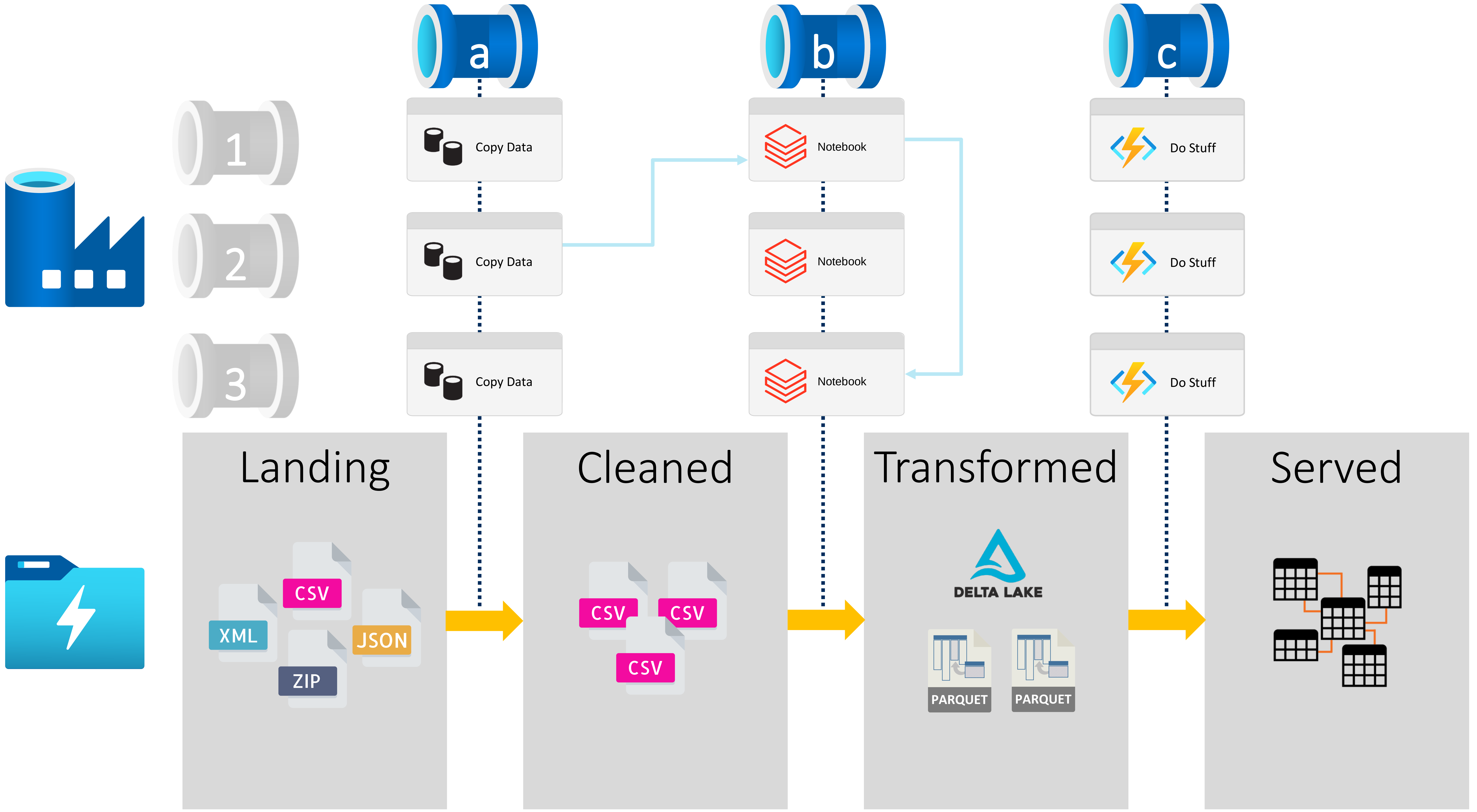
Cloud Formations - Knowledge Transfer & Training

© 2024 Cloud Formations Ltd

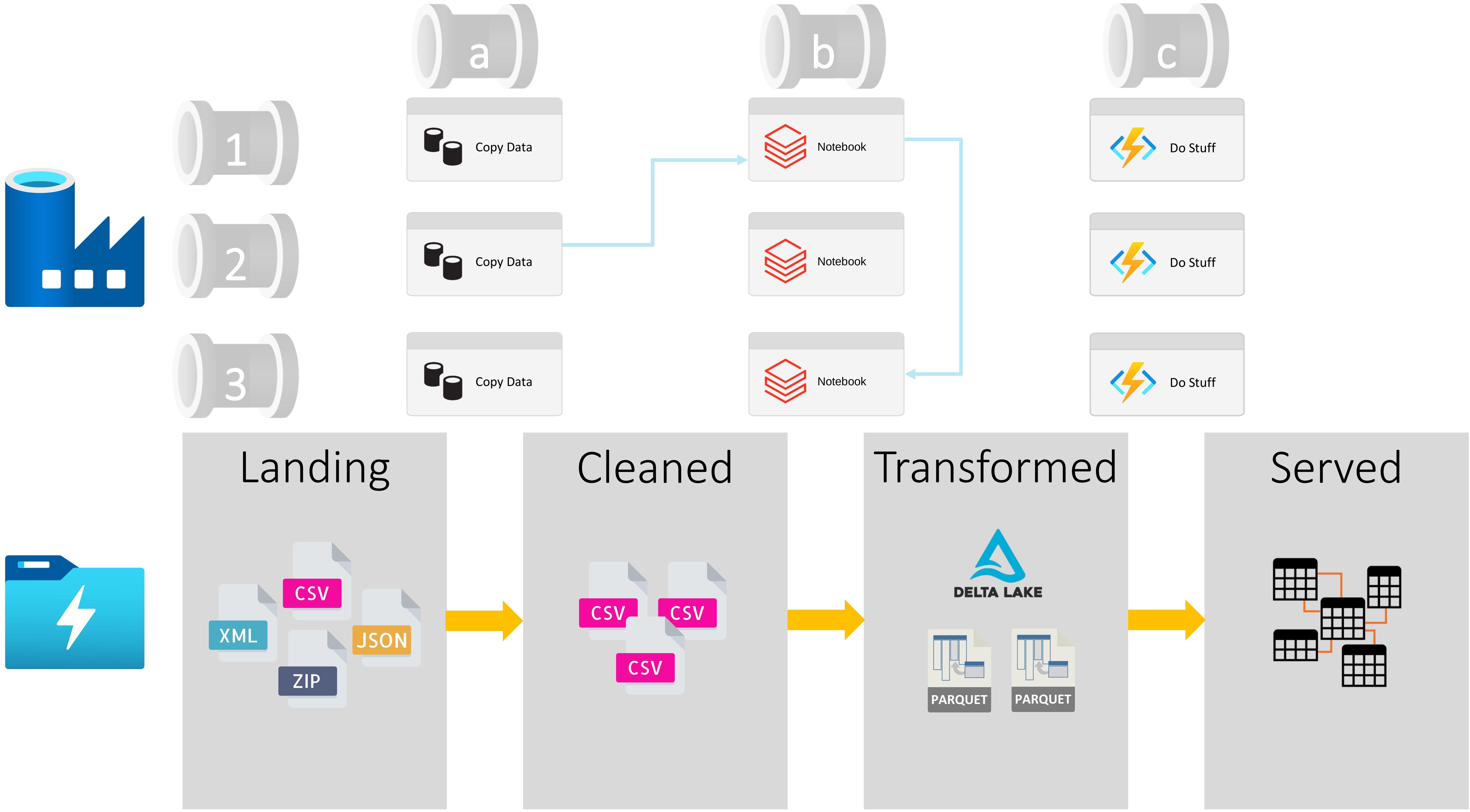
Problem


a
b
c
1
2
3
Copy Data
Copy Data
Copy Data
Notebook
Notebook
Notebook
Do Stuff
Do Stuff
Do Stuff
Landing
CSV
XML
JSON
ZIP
Cleaned
CSV
CSV
CSV
Transformed
DELTA LAKE
PARQUET
PARQUET
Served

Problem


a
b
c
1
2
3
Copy Data
Copy Data
Copy Data
Notebook
Notebook
Notebook
Do Stuff
Do Stuff
Do Stuff
Landing
CSV
XML
JSON
ZIP
Cleaned
CSV
CSV
CSV
Transformed
DELTA LAKE
PARQUET
PARQUET
Served

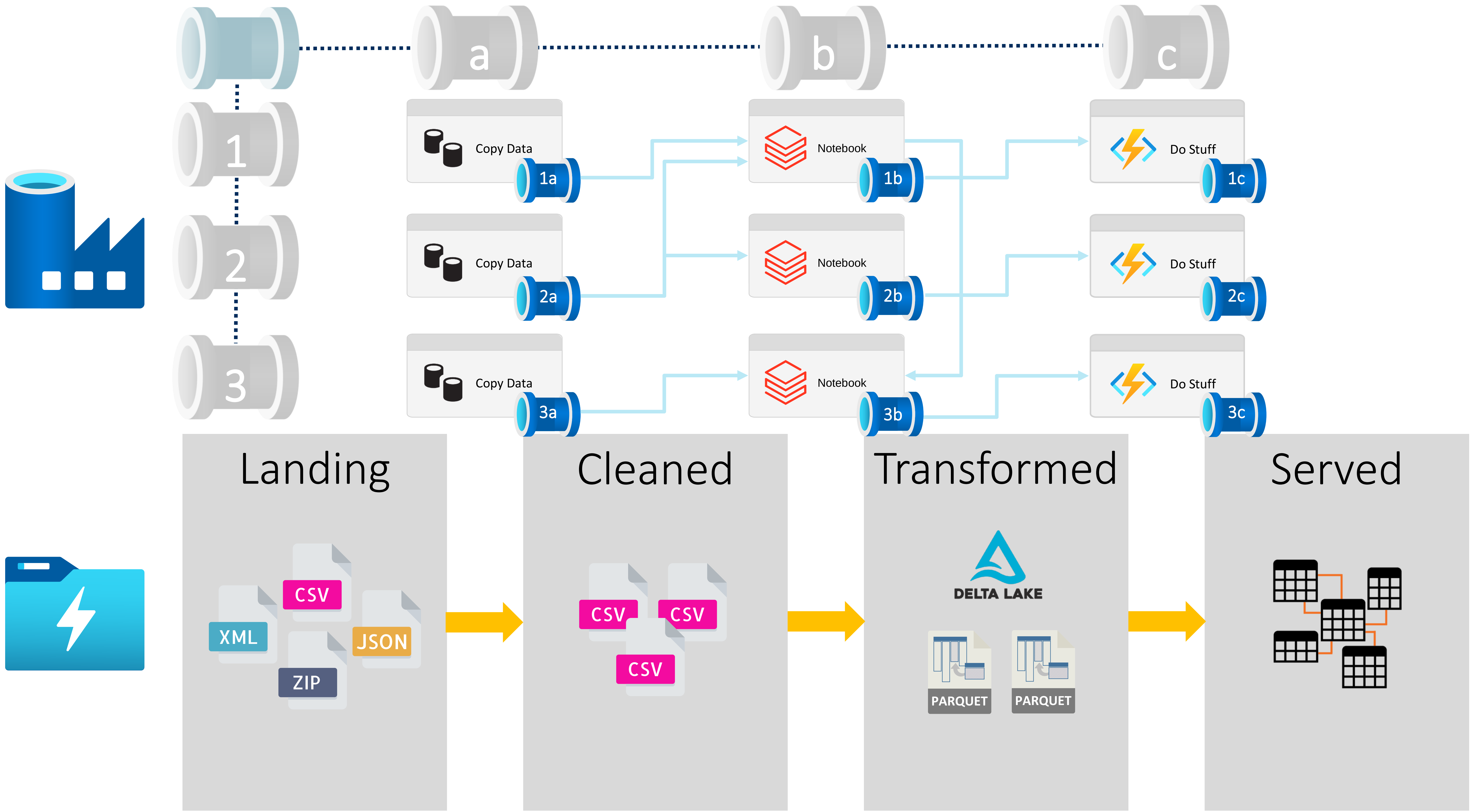
Problem



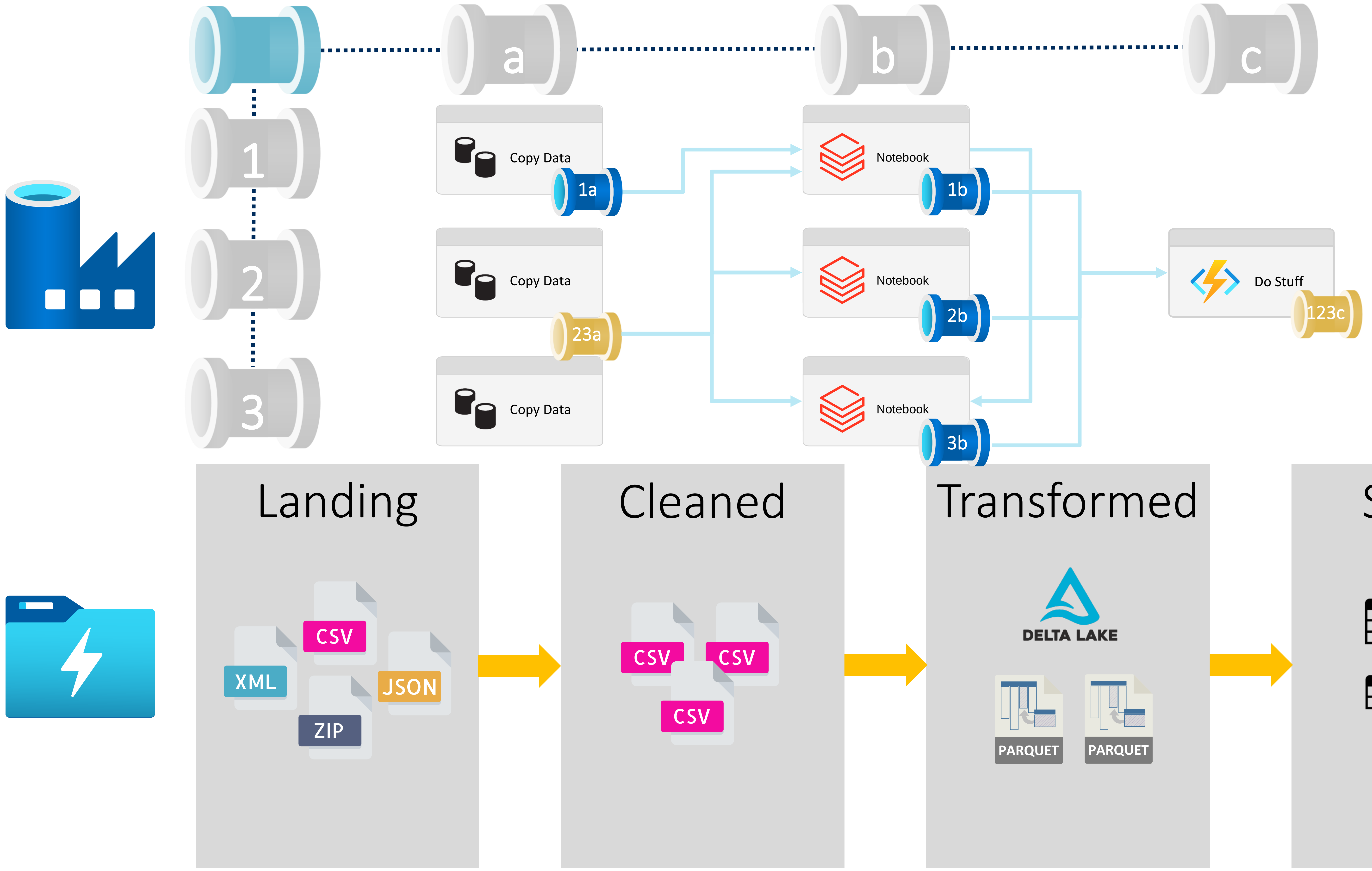
Cloud Formations - Knowledge Transfer & Training

Problem

Only 80 Activities per Pipeline.



Problem



Grandparent pipeline for all processing.

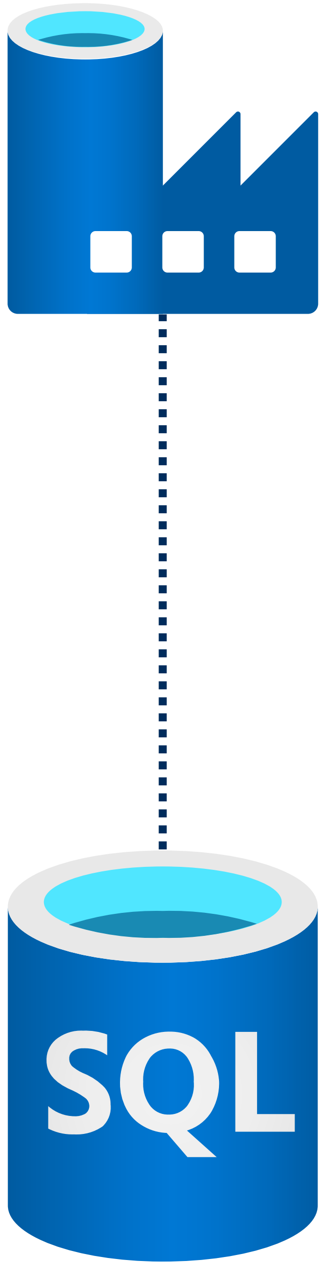
Parent pipeline to consolidate work.

Child pipelines for low level dependencies.

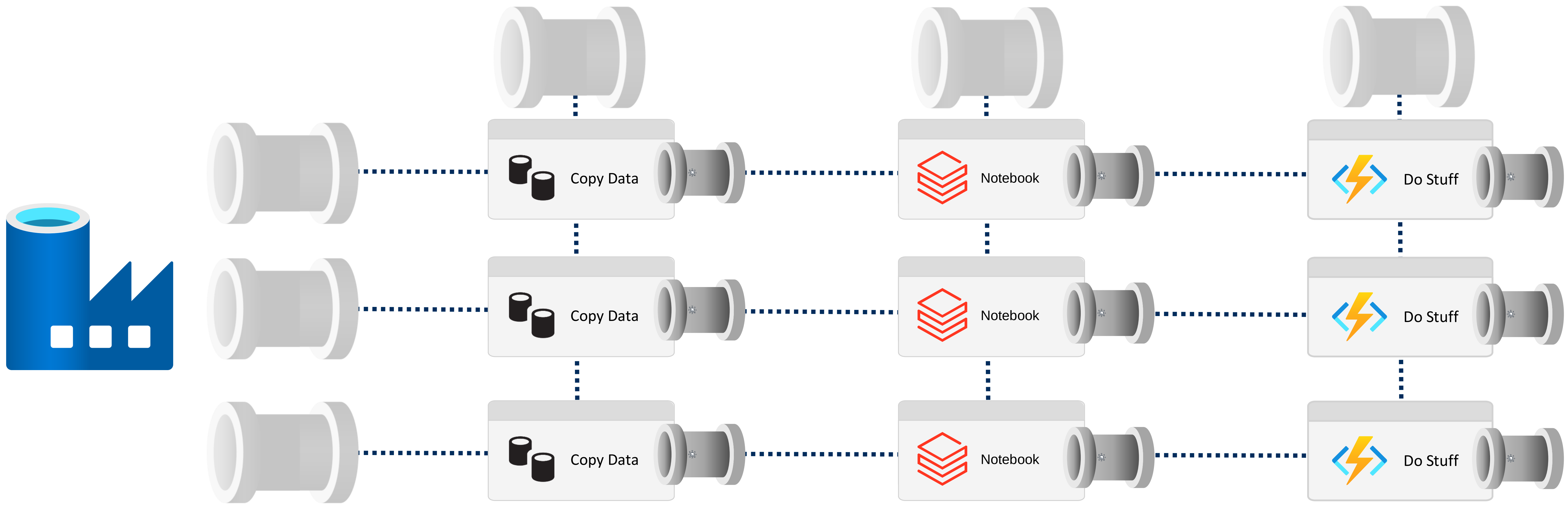
Cloud Formations - Knowledge Transfer & Training

Solution

Use metadata to drive all pipeline executions



Solution



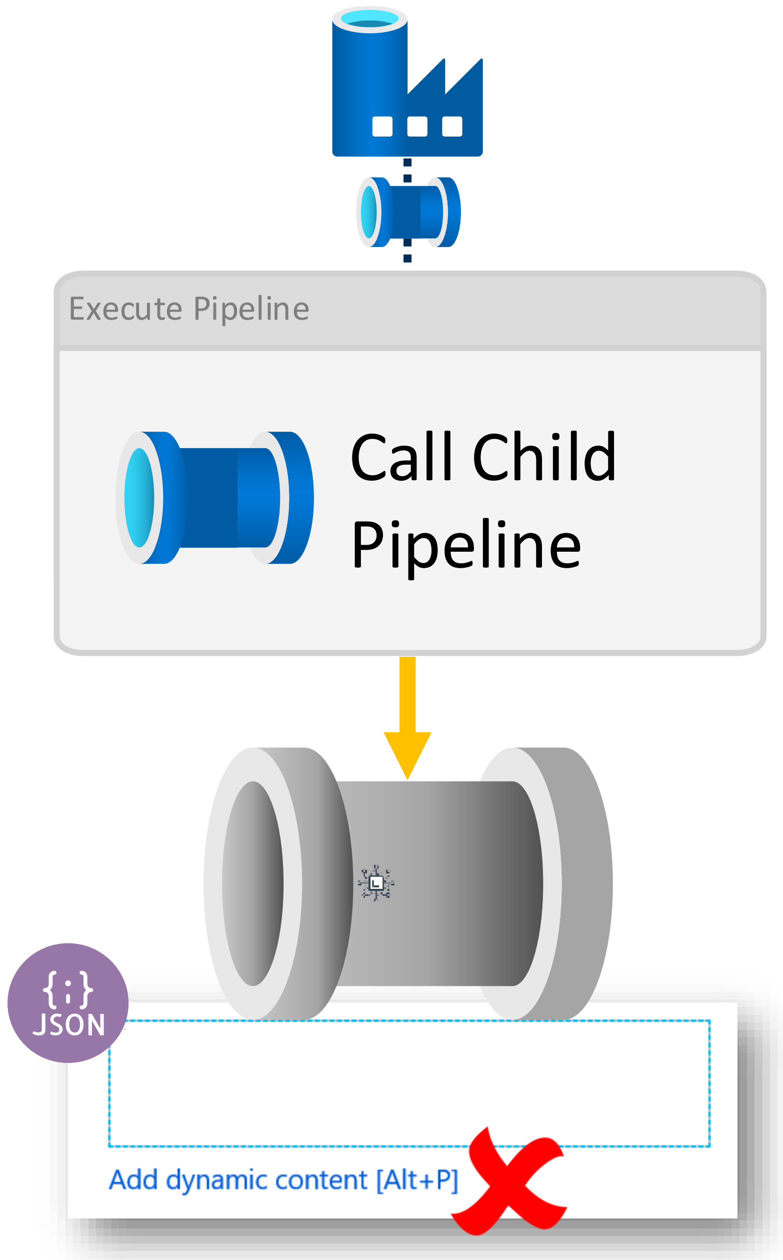
Stages
1
2
3

Pipelines
a
b
c
d
e
f
g
h
i

Stage	Pipeline
1	a
1	b
1	c
2	d
2	e
3	f
3	g
3	h
3	i

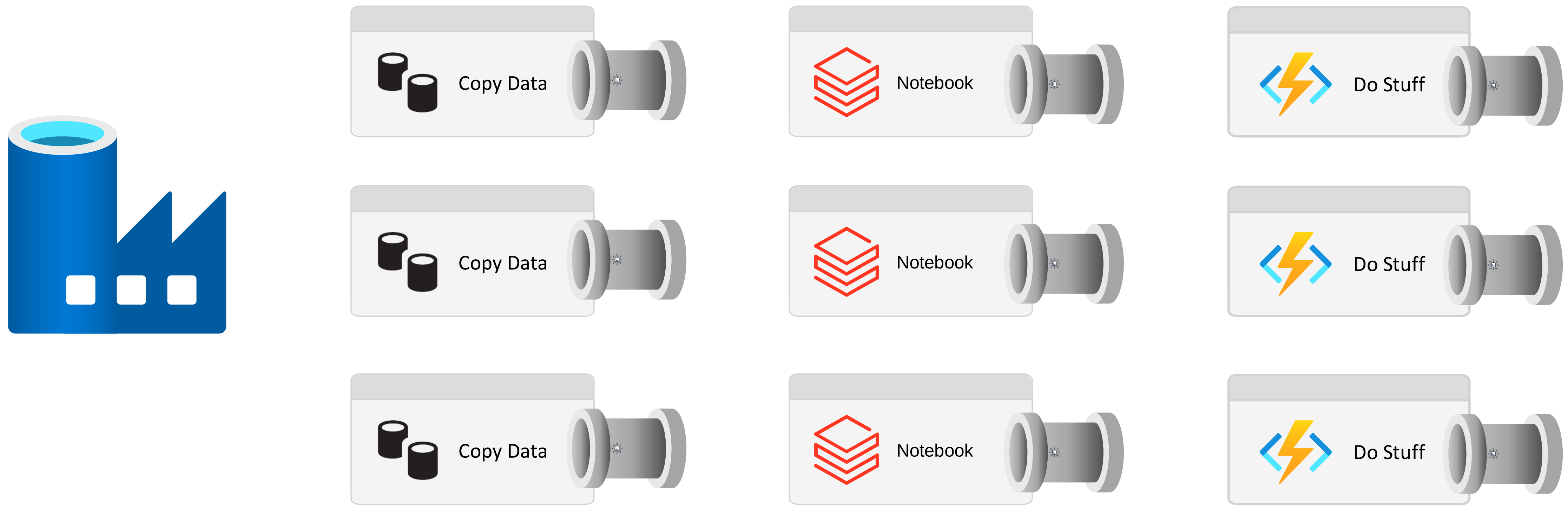
Calling Our Worker Pipelines

One More Problem to Consider



Calling Our Worker Pipelines

Cloud Formations - Knowledge Transfer & Training

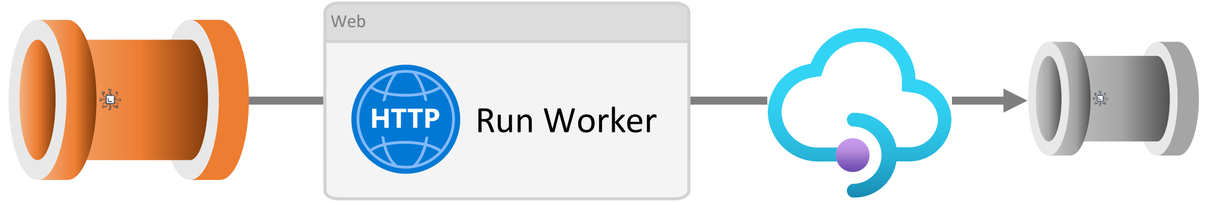


Stages
1
2
3

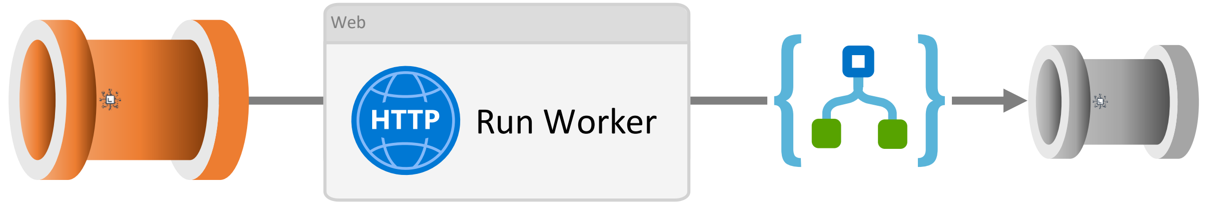
Pipelines
a
b
c
d
e
f
g
h
i

Stage	Pipeline
1	a
1	b
1	c
2	d
2	e
3	f
3	g
3	h
3	i

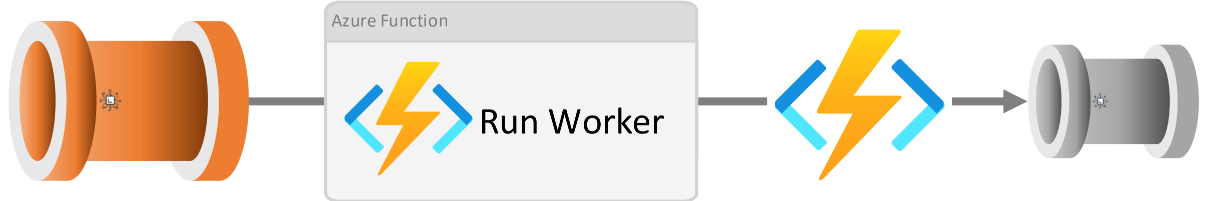
Option 1:



Option 2:

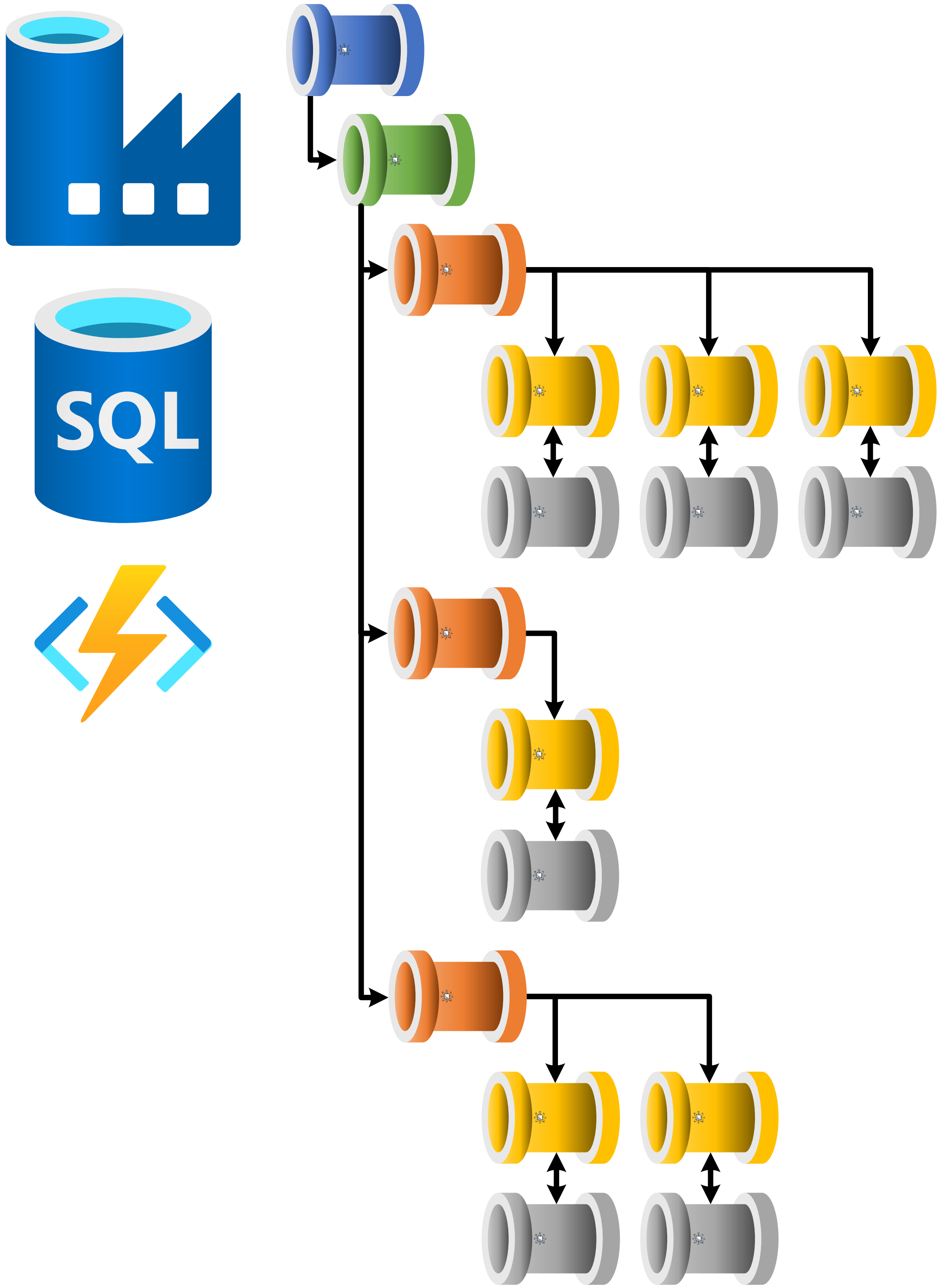


Option 3:

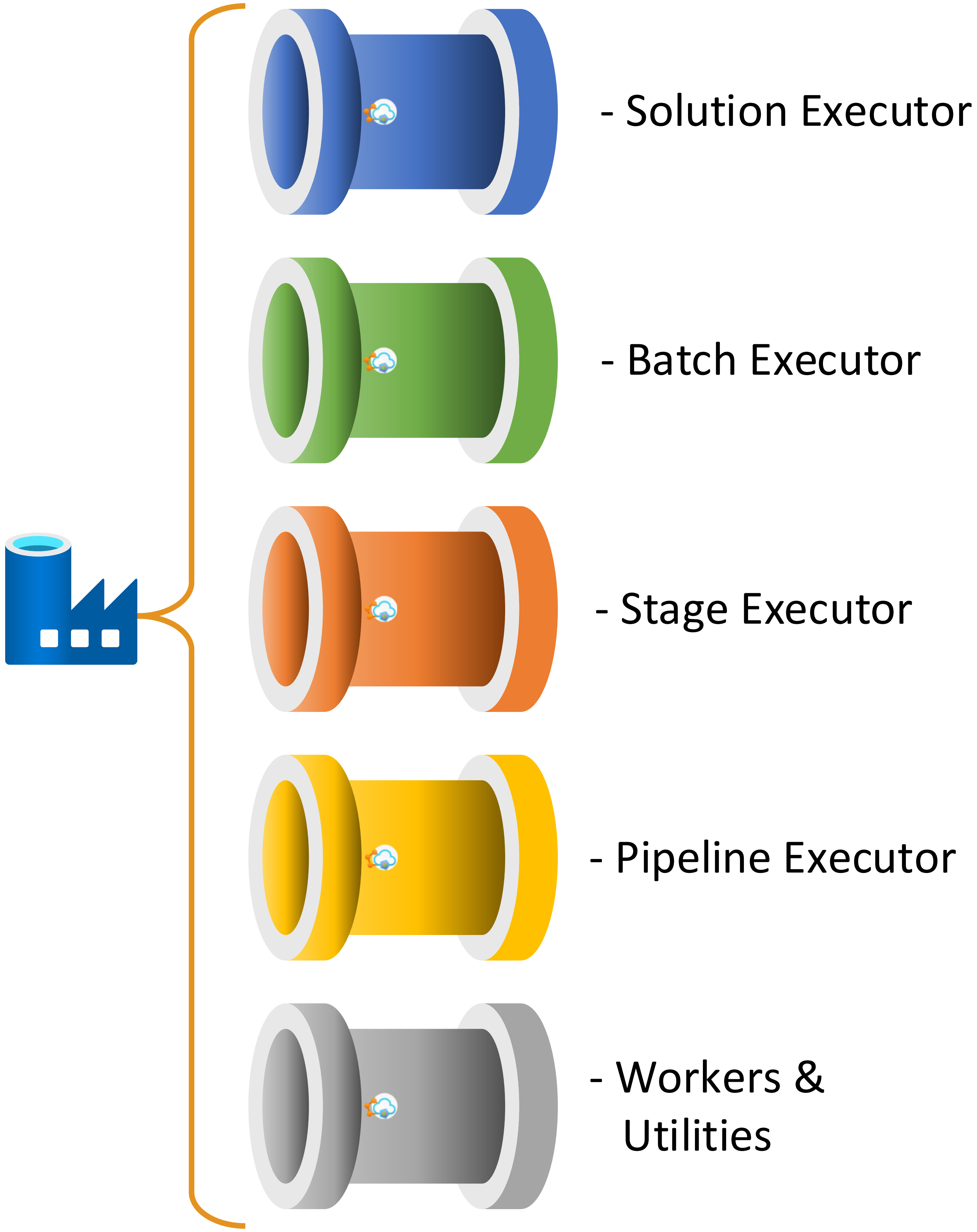


Solution

Use Metadata to Drive Data Factory Pipelines & Functions as the middleware to handle worker execution



Framework Pipeline Hierarchy



- Solution Executor

Role: Optional level platform setup, for example, scale up/out compute services ready for the framework to run.

- Batch Executor

Role: Execution run wrapper for batches and execution stage iterator.

- Stage Executor

Role: Scale out triggering of worker pipelines within the execution stage(s).

- Pipeline Executor

Role: Worker validator, executor, monitor and reporting of the outcome for the single worker pipeline.

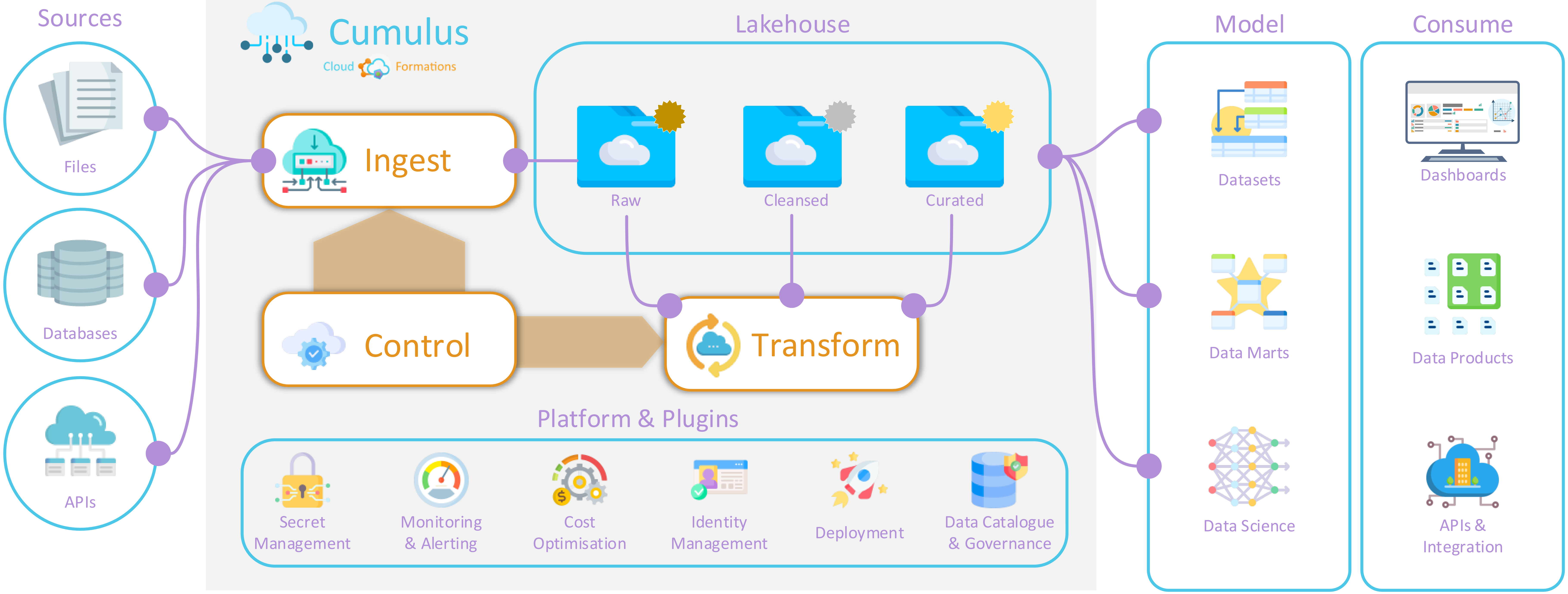
- Workers & Utilities

Role: Anything specific to the process needing to be performed, Ingest and Transform pipelines included.

Introducing CF.Cumulus

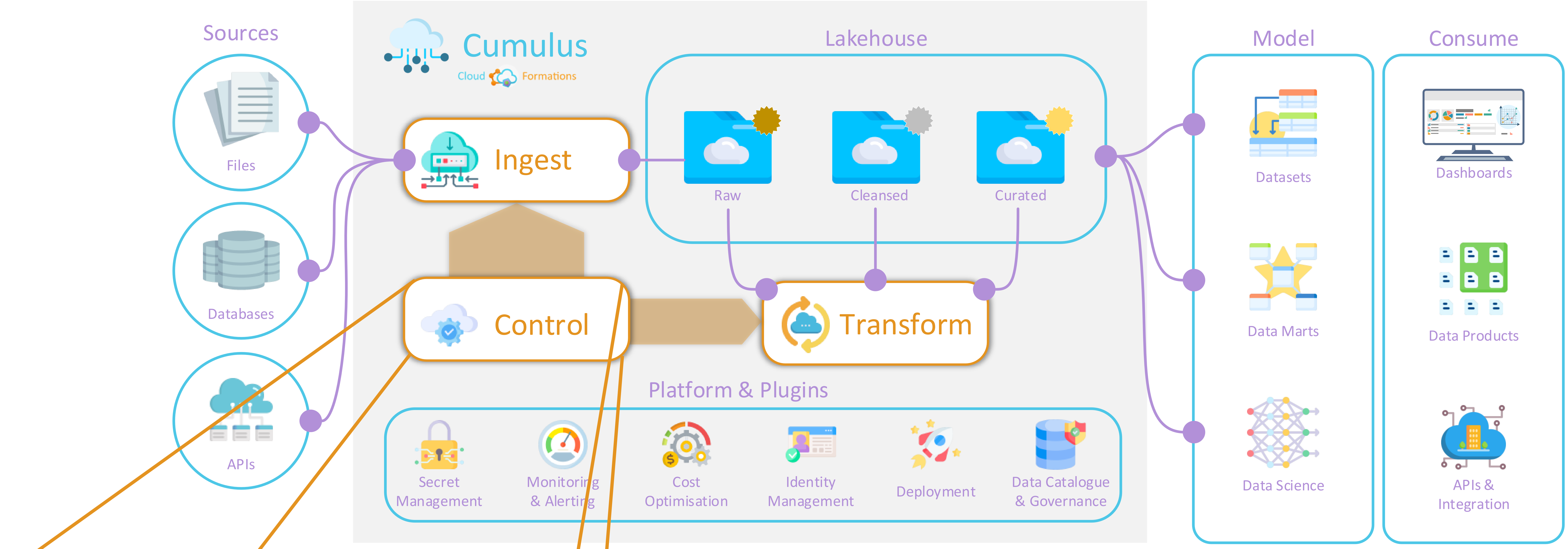
cloudformations.org/cumulus

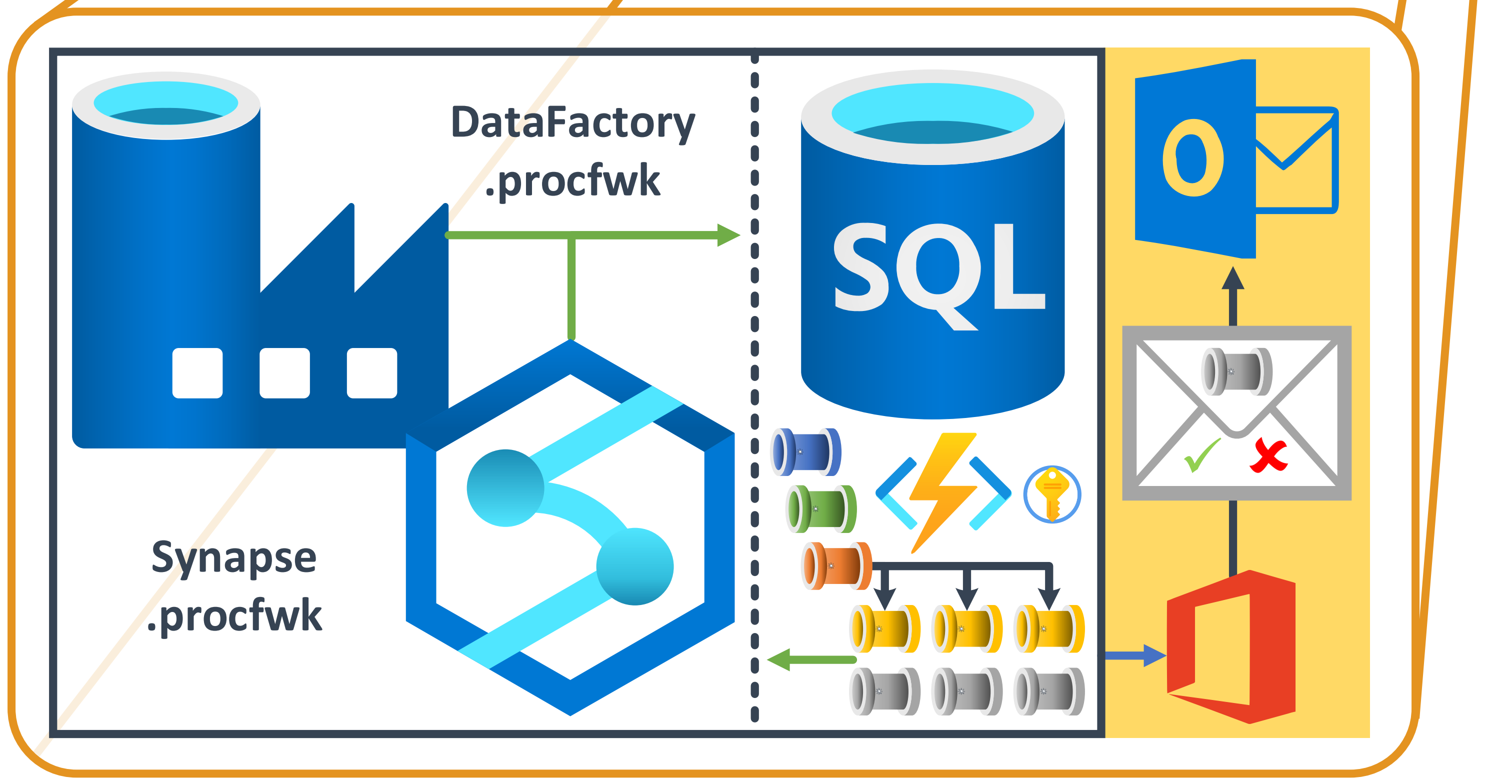
A cloud data platform product to accelerate time to insights. Our open-source framework is designed for the real world. Stripping away the complexity, giving you the power to build, scale, and manage your dataflows with ease, accelerating data delivery.



Underlying Technology: Spark, DELTA LAKE, C#, JSON

Works With:





ProcFwk.com

An evolution from metadata driven orchestration to complete data delivery. Accelerating your time to insight. And still open-source.

Cumulus

Cloud Formations

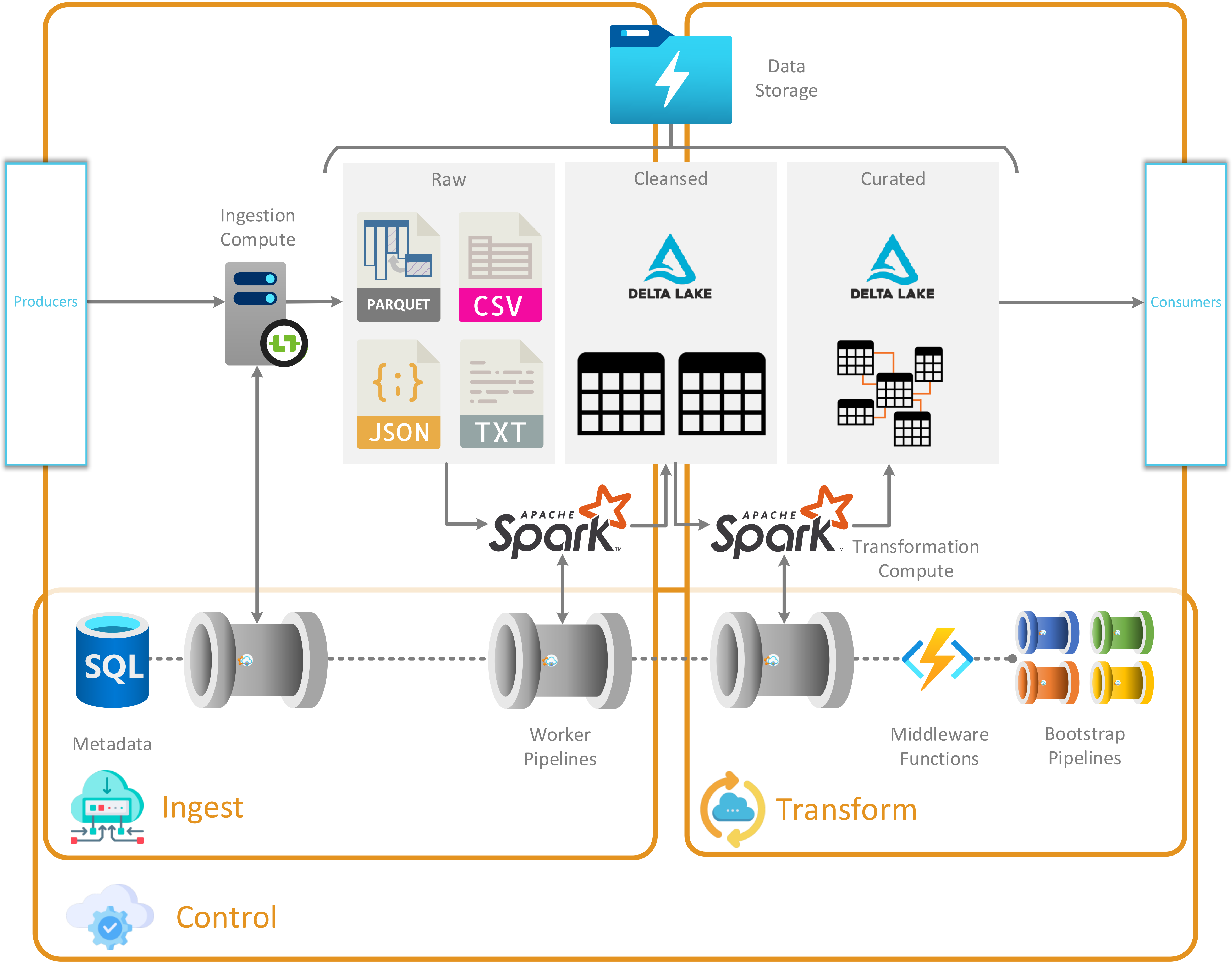
CF.Cumulus Framework

Cloud Formations - Logical Architecture

An Azure product agnostic view of compute, storage and orchestration resources as used by CF.Cumulus.

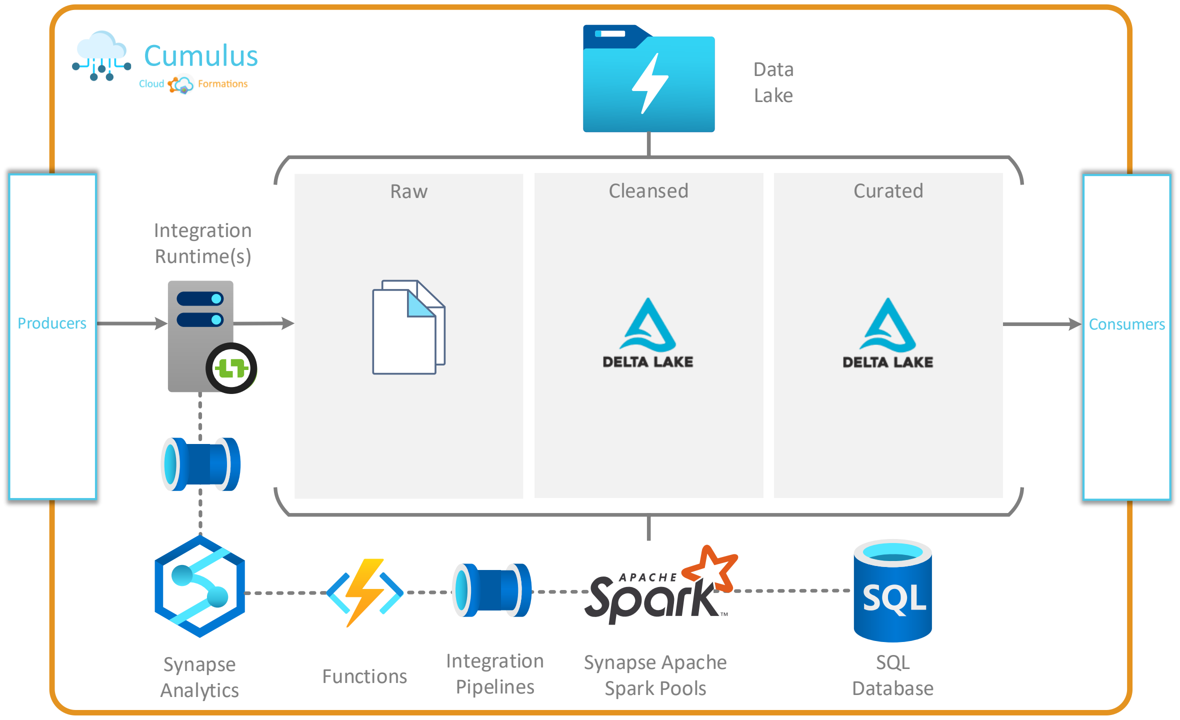
A primary design focus for CF.Cumulus is to align with open-standards and open technologies. Offering a flexible, plug and play approach to the core components.

In this context, using Apache Spark compute and Delta Lake structures for the storage layer.



Flexible Technology Choices

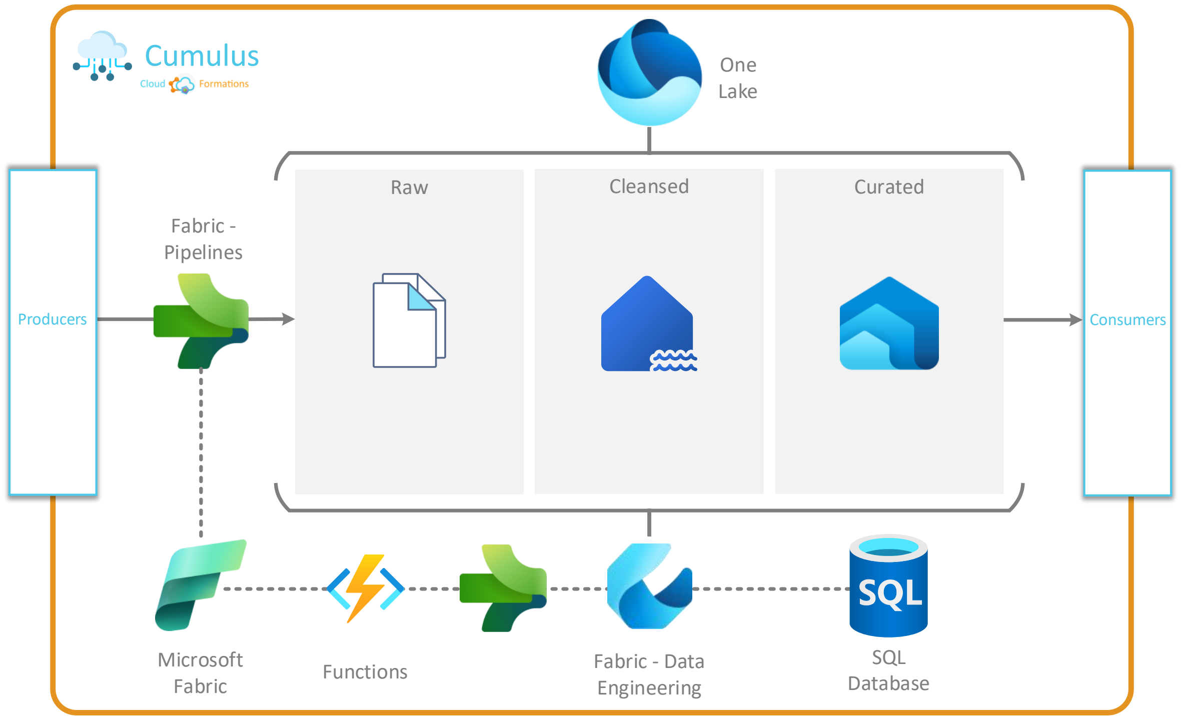
Building on the generic visuals above, but now applying a pure Microsoft product perspective to the logical delivery of Cumulus. The following diagrams could apply to a solution depending on the choice of tooling for Orchestration, Compute and Storage with each offering different advantages and coupling of resources.



Using Azure Synapse Analytics

This implementation represents a partly decoupled architecture, in terms of cloud resources used to delivery Cumulus.

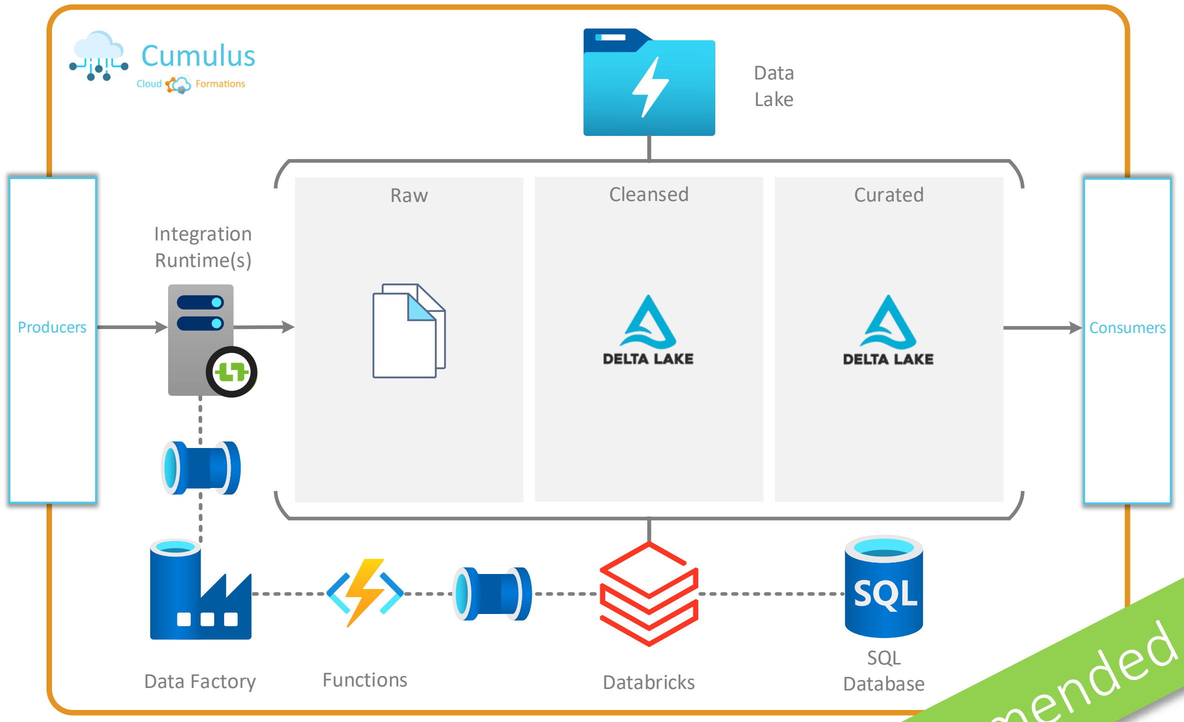
- **Orchestration** and **Compute** delivered using Azure Synapse Analytics.
- **Storage** delivered using Azure Data Lake.



Using Microsoft Fabric

This implementation represents a coupled architecture, in terms of cloud resources used to delivery Cumulus.

- **Orchestration, Compute** and **Storage** delivered using Microsoft Fabric experiences.



Recommended

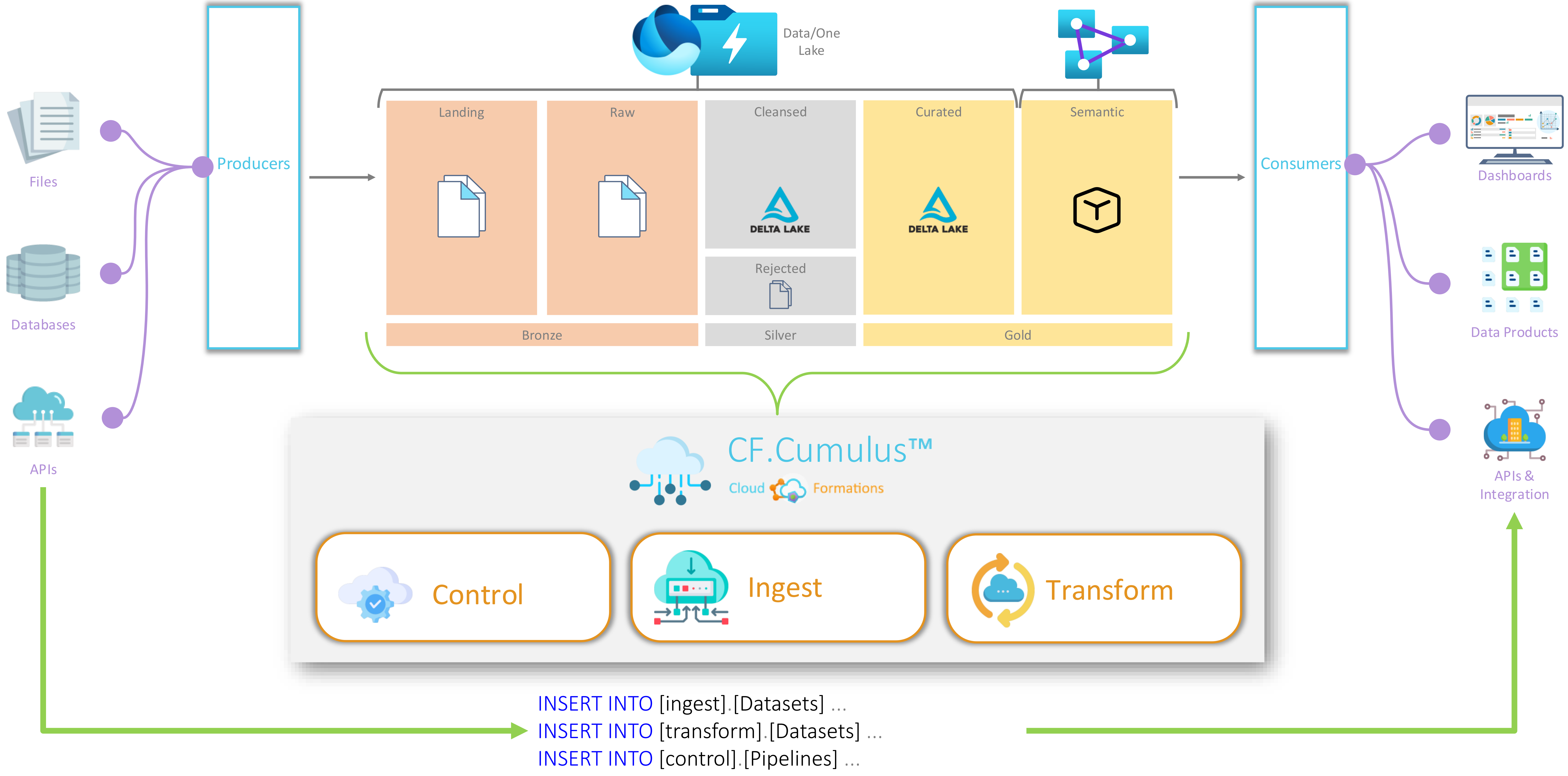
Using Azure Data Factory & Azure Databricks

This implementation represents a fully decoupled architecture, in terms of cloud resources used to delivery Cumulus.

- **Orchestration** delivered using Azure Data Factory.
- **Compute** delivered using Azure Databricks.
- **Storage** delivered using Azure Data Lake.

Lakehouse Implementations - Simplified Through Metadata

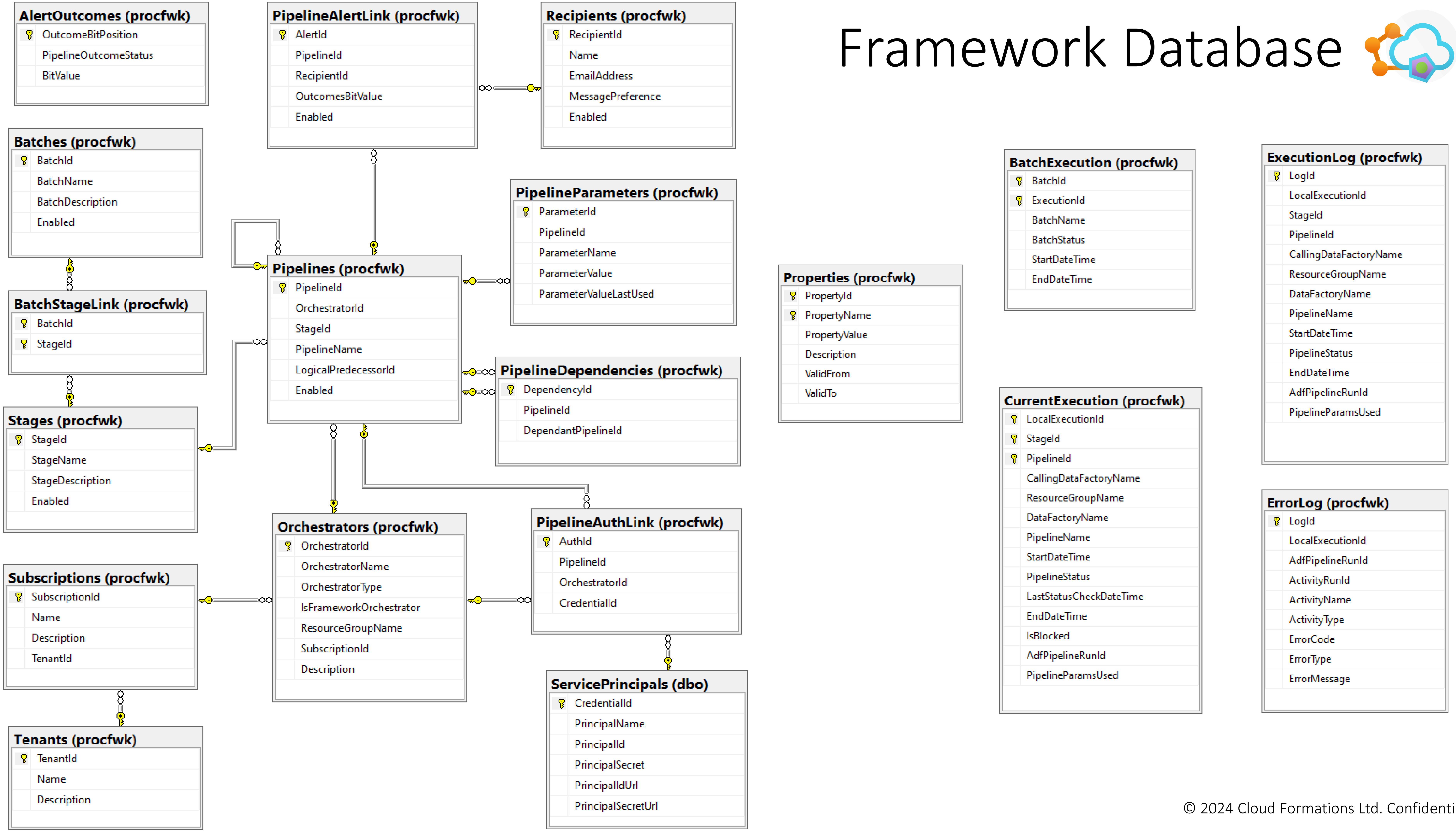
New source datasets delivered from producers to consumers with ease using only a few lines of metadata and open cloud native tooling.



```
INSERT INTO [ingest].[Datasets] ...
INSERT INTO [transform].[Datasets] ...
INSERT INTO [control].[Pipelines] ...
```

Cloud Formations

Framework Database



AlertOutcomes (procfwk)
- OutcomeBitPosition
- PipelineOutcomeStatus
- BitValue

PipelineAlertLink (procfwk)
- AlertId
- PipelineId
- RecipientId
- OutcomesBitValue
- Enabled

Recipients (procfwk)
- RecipientId
- Name
- EmailAddress
- MessagePreference
- Enabled

Batches (procfwk)
- BatchId
- BatchName
- BatchDescription
- Enabled

BatchStageLink (procfwk)
- BatchId
- StageId

Stages (procfwk)
- StageId
- StageName
- StageDescription
- Enabled

Pipelines (procfwk)
- PipelineId
- OrchestratorId
- StageId
- PipelineName
- LogicalPredecessorId
- Enabled

PipelineParameters (procfwk)
- ParameterId
- PipelineId
- ParameterName
- ParameterValue
- ParameterValueLastUsed

PipelineDependencies (procfwk)
- DependencyId
- PipelineId
- DependantPipelineId

Properties (procfwk)
- PropertyId
- PropertyName
- PropertyValue
- Description
- ValidFrom
- ValidTo

BatchExecution (procfwk)
- BatchId
- ExecutionId
- BatchName
- BatchStatus
- StartDateTime
- EndDateTime

ExecutionLog (procfwk)
- LogId
- LocalExecutionId
- StageId
- PipelineId
- CallingDataFactoryName
- ResourceGroupName
- DataFactoryName
- PipelineName
- StartDateTime
- PipelineStatus
- EndDateTime
- AdfPipelineRunId
- PipelineParamsUsed

Subscriptions (procfwk)
- SubscriptionId
- Name
- Description
- TenantId

Tenants (procfwk)
- TenantId
- Name
- Description

Orchestrators (procfwk)
- OrchestratorId
- OrchestratorName
- OrchestratorType
- IsFrameworkOrchestrator
- ResourceGroupName
- SubscriptionId
- Description

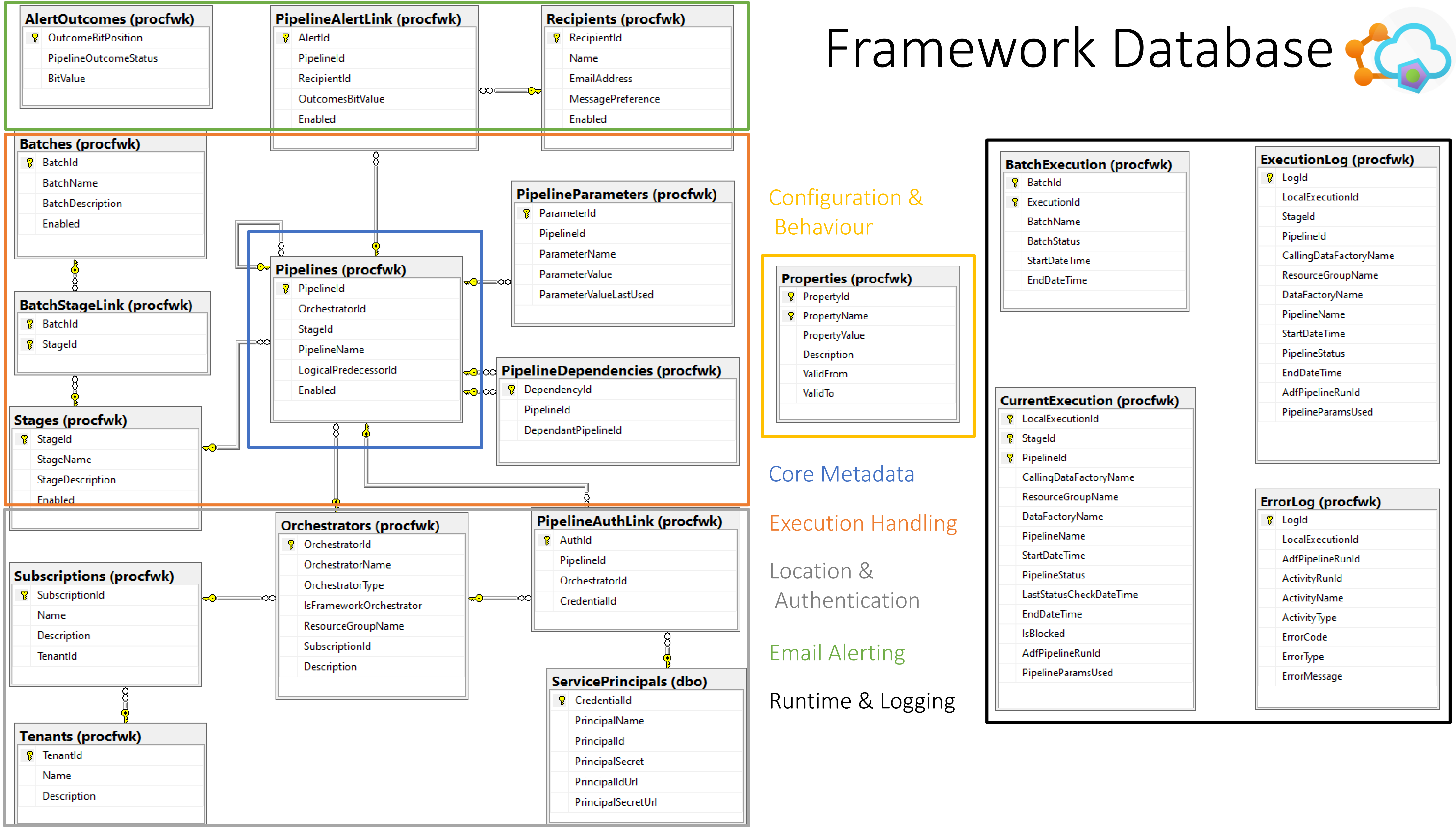
PipelineAuthLink (procfwk)
- AuthId
- PipelineId
- OrchestratorId
- CredentialId

ServicePrincipals (dbo)
- CredentialId
- PrincipalName
- PrincipalId
- PrincipalSecret
- PrincipalIdUrl
- PrincipalSecretUrl

CurrentExecution (procfwk)
- LocalExecutionId
- StageId
- PipelineId
- CallingDataFactoryName
- ResourceGroupName
- DataFactoryName
- PipelineName
- StartDateTime
- PipelineStatus
- LastStatusCheckDateTime
- EndDateTime
- IsBlocked
- AdfPipelineRunId
- PipelineParamsUsed

ErrorLog (procfwk)
- LogId
- LocalExecutionId
- AdfPipelineRunId
- ActivityRunId
- ActivityName
- ActivityType
- ErrorCode
- ErrorType
- ErrorMessage

Framework Database



Configuration & Behaviour

Core Metadata

Execution Handling

Location & Authentication

Email Alerting

Runtime & Logging

AlertOutcomes (procfwk)
- OutcomeBitPosition
- PipelineOutcomeStatus
- BitValue

PipelineAlertLink (procfwk)
- AlertId
- PipelineId
- RecipientId
- OutcomesBitValue
- Enabled

Recipients (procfwk)
- RecipientId
- Name
- EmailAddress
- MessagePreference
- Enabled

Batches (procfwk)
- BatchId
- BatchName
- BatchDescription
- Enabled

BatchStageLink (procfwk)
- BatchId
- StageId

Stages (procfwk)
- StageId
- StageName
- StageDescription
- Enabled

Pipelines (procfwk)
- PipelineId
- OrchestratorId
- StageId
- PipelineName
- LogicalPredecessorId
- Enabled

PipelineParameters (procfwk)
- ParameterId
- PipelineId
- ParameterName
- ParameterValue
- ParameterValueLastUsed

PipelineDependencies (procfwk)
- DependencyId
- PipelineId
- DependantPipelineId

Properties (procfwk)
- PropertyId
- PropertyName
- PropertyValue
- Description
- ValidFrom
- ValidTo

Subscriptions (procfwk)
- SubscriptionId
- Name
- Description
- TenantId

Tenants (procfwk)
- TenantId
- Name
- Description

Orchestrators (procfwk)
- OrchestratorId
- OrchestratorName
- OrchestratorType
- IsFrameworkOrchestrator
- ResourceGroupName
- SubscriptionId
- Description

PipelineAuthLink (procfwk)
- AuthId
- PipelineId
- OrchestratorId
- CredentialId

ServicePrincipals (dbo)
- CredentialId
- PrincipalName
- PrincipalId
- PrincipalSecret
- PrincipalIdUrl
- PrincipalSecretUrl

BatchExecution (procfwk)
- BatchId
- ExecutionId
- BatchName
- BatchStatus
- StartDateTime
- EndDateTime

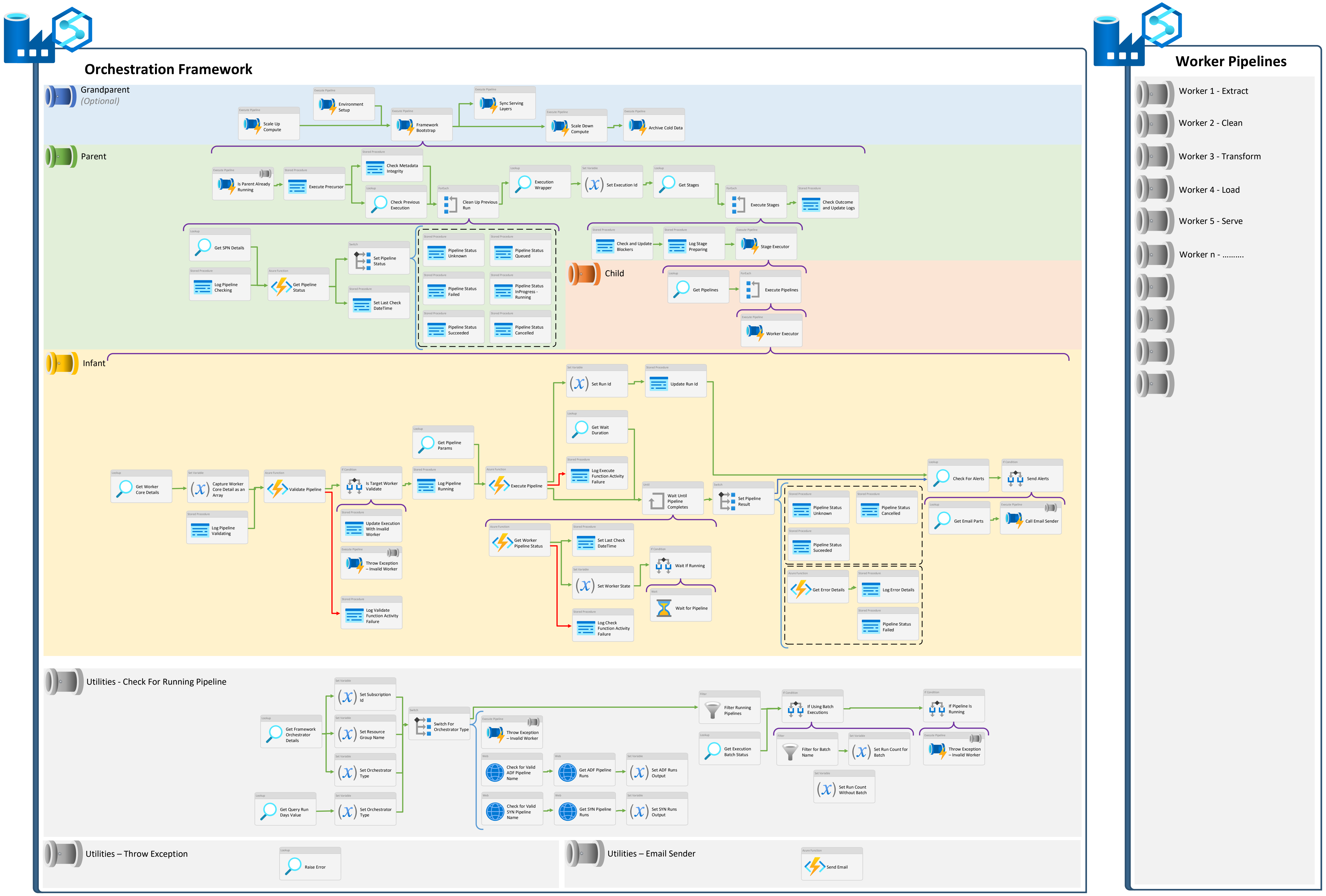
CurrentExecution (procfwk)
- LocalExecutionId
- StageId
- PipelineId
- CallingDataFactoryName
- ResourceGroupName
- DataFactoryName
- PipelineName
- StartDateTime
- PipelineStatus
- LastStatusCheckDateTime
- EndDateTime
- IsBlocked
- AdfPipelineRunId
- PipelineParamsUsed

ExecutionLog (procfwk)
- LogId
- LocalExecutionId
- StageId
- PipelineId
- CallingDataFactoryName
- ResourceGroupName
- DataFactoryName
- PipelineName
- StartDateTime
- PipelineStatus
- EndDateTime
- AdfPipelineRunId
- PipelineParamsUsed

ErrorLog (procfwk)
- LogId
- LocalExecutionId
- AdfPipelineRunId
- ActivityRunId
- ActivityName
- ActivityType
- ErrorCode
- ErrorType
- ErrorMessage

CF.Cumulus.Control Activity Chain

Cloud Formations - Knowledge Transfer & Training



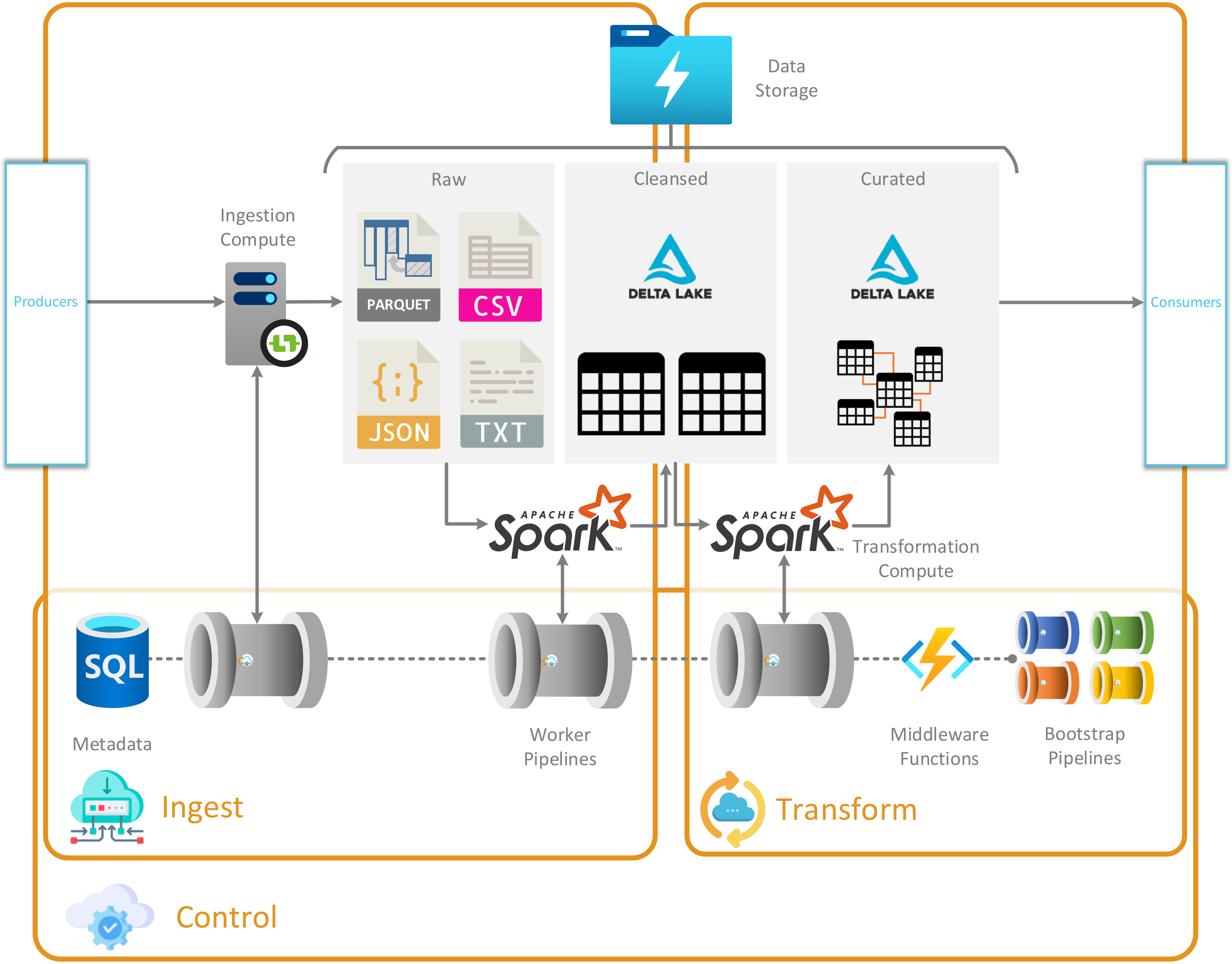
CF.Cumulus Framework

Cloud Formations - Logical Architecture

An Azure product agnostic view of compute, storage and orchestration resources as used by CF.Cumulus.

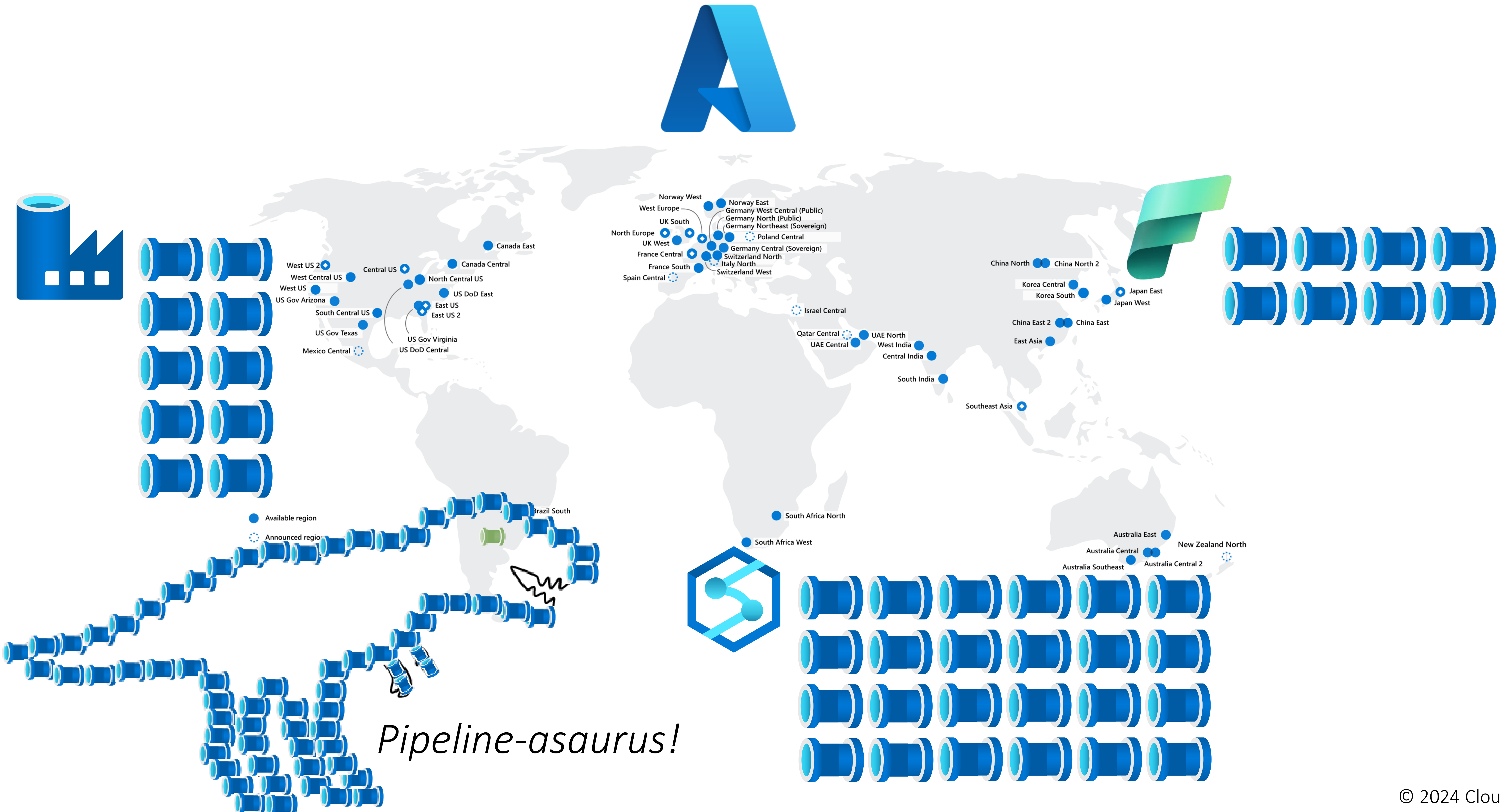
A primary design focus for CF.Cumulus is to align with open-standards and open technologies. Offering a flexible, plug and play approach to the core components.

In this context, using Apache Spark compute and Delta Lake structures for the storage layer.



Problem

How should we structure and trigger our Integration Pipelines?

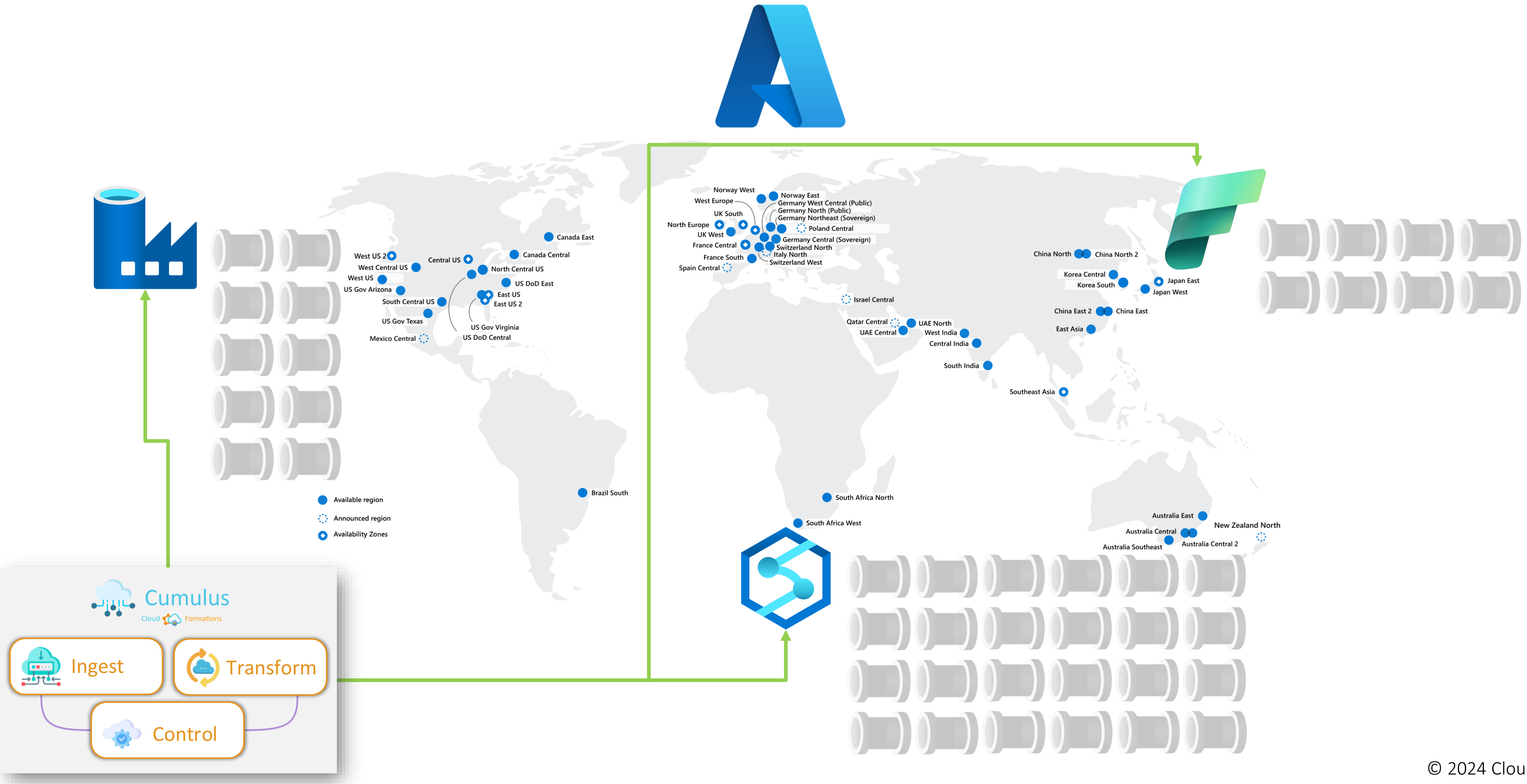


Pipeline-asaurus!

Solution

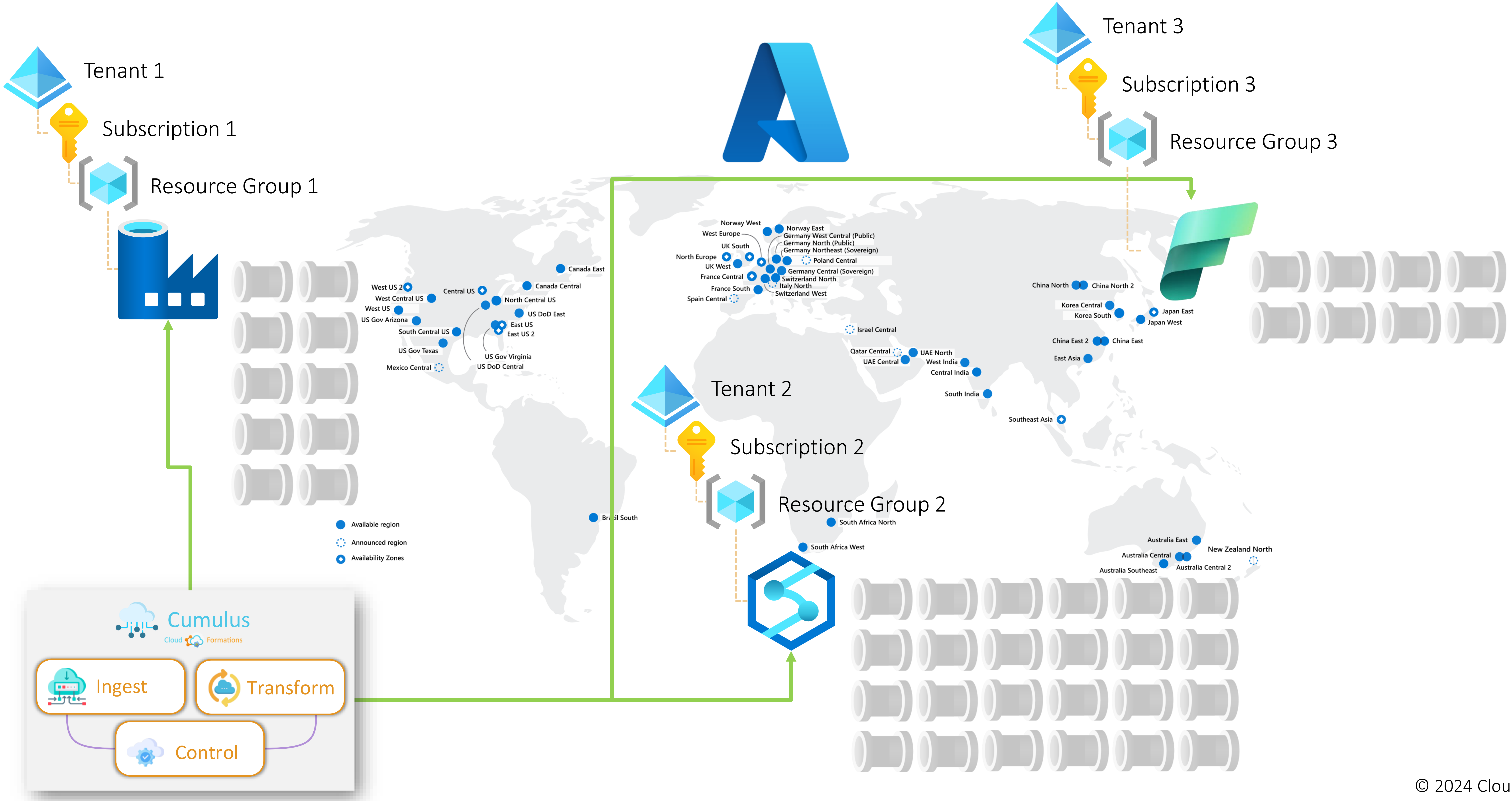
Cloud Formations - Knowledge Transfer & Training

Use Metadata to Drive Integration Pipeline execution



Solution

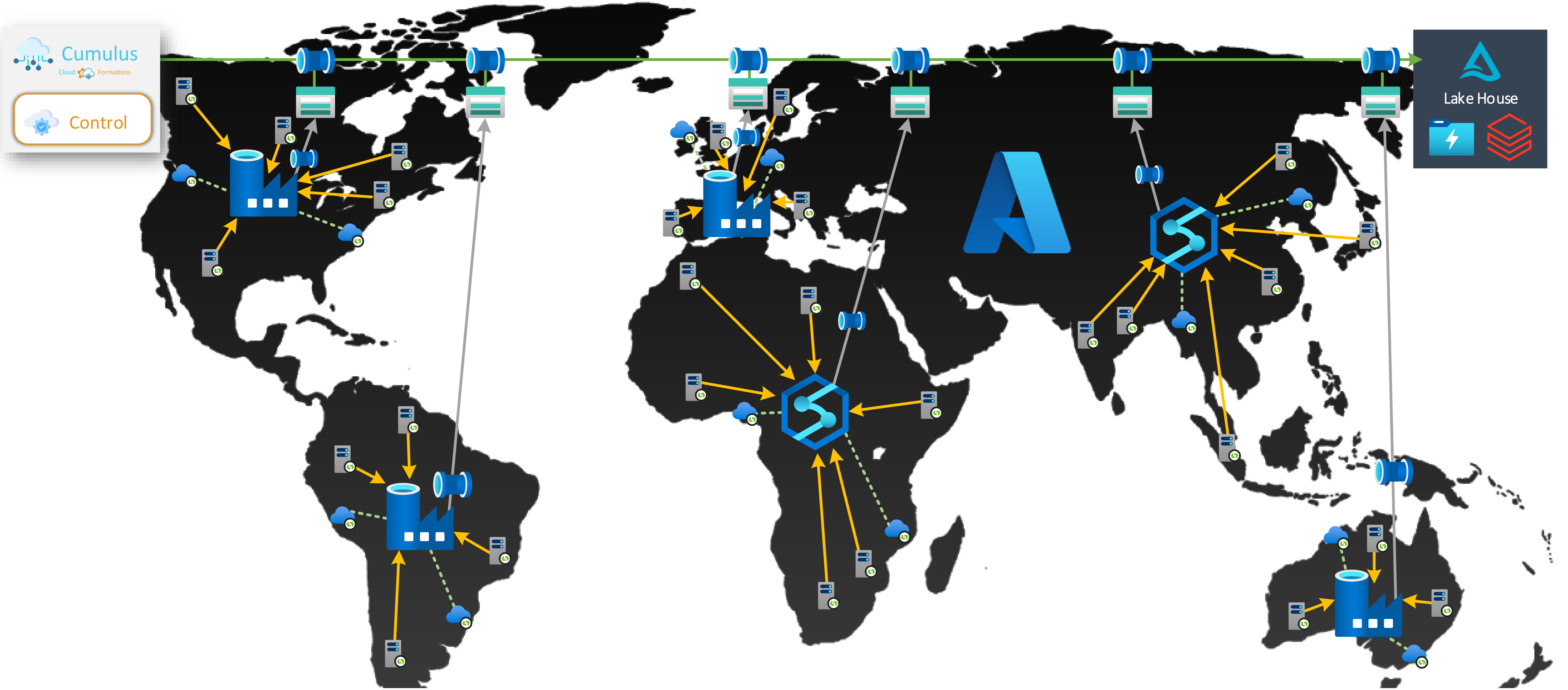
Use Metadata to Drive Integration Pipeline execution



Cloud Formations - Knowledge Transfer & Training

Hub & Spoke Integration Architecture

Cloud Formations - Knowledge Transfer & Training

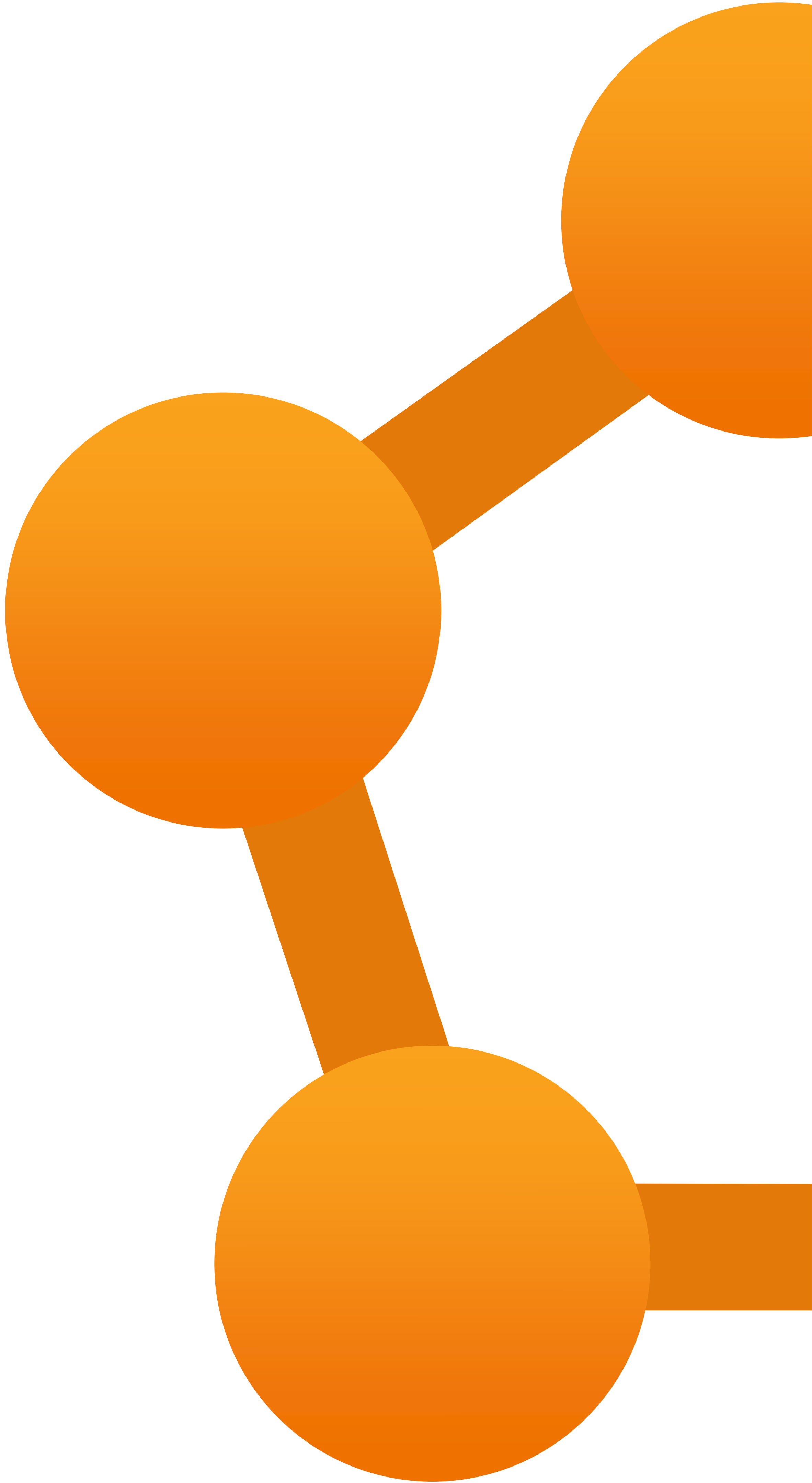


https://mrpaulandrew.com/2021/08/10/scaling-azure-data-integration-pipelines-decoupling-data-extract-and-transform/

Module 6

Execution Parallelism ✓

Any questions?



Cloud Formations

Agenda: Data Integration Pipelines

Cloud Formations - Knowledge Transfer & Training

Fundamentals to Level 300

- Module 1: Pipeline Fundamentals
  - The History of Azure Orchestration
  - Synapse Analytics vs Data Factory vs Microsoft Fabric
  - Integration Components
  - Common Activities
  - Execution Dependencies
- Module 2: Integration Runtime Design Patterns
  - Compute Types
    - Azure
    - Hosted
    - SSIS
  - Patterns & Configuration
- Module 3: Data Transformation
  - Data Flows
  - Power Query Injection
  - Spark Configuration
  - Use Cases

<< BREAK

- Module 4: Dynamic Pipelines
  - Expressions & Interpolation
  - Simple Metadata Driven Execution
  - Dynamic Content Chains
  - Reference Names
- Module 5: Pipeline Extensibility
  - Azure Batch Service
  - Pipeline Custom Activities
  - Azure Management API
  - Azure Functions
- Labs
  - Create Azure resources
  - Build a copy pipeline
  - Create a reusable pipeline
  - Author a data flow
  - Monitor factory activity
  - Explore Synapse pipelines
  - Explore Fabric pipelines
  - Mini-project

<< LUNCH

- Module 6: Execution Parallelism
  - Control Flow Scale Out
  - Concurrency Limitations
  - Internal vs External Activities
  - Orchestration Framework
- Module 7: VNet Integration
  - Private Endpoints
  - Managed VNet's
  - Firewall Bypass
- Module 8: Security
  - Service Principals
  - Managed Identities
  - Azure Key Vault Integration
  - Customer Managed Keys
  - Pipeline Access & Permissions

<< BREAK

- Module 9: Monitoring & Alerting
  - Studio Monitoring
  - Log Analytics & Kusto Queries
  - Operational Dashboards
  - Advanced Alerting
- Module 10: Solution Testing
  - Development Time Validation
  - Test Coverage
  - NUnit Tests
- Module 11: CI/CD
  - Source Control vs Developer UI
  - Basic ARM Template Deployments
  - Advanced Deployment Patterns
- Module 12: Final Thoughts
  - Costs & Conclusions
  - Best Practices