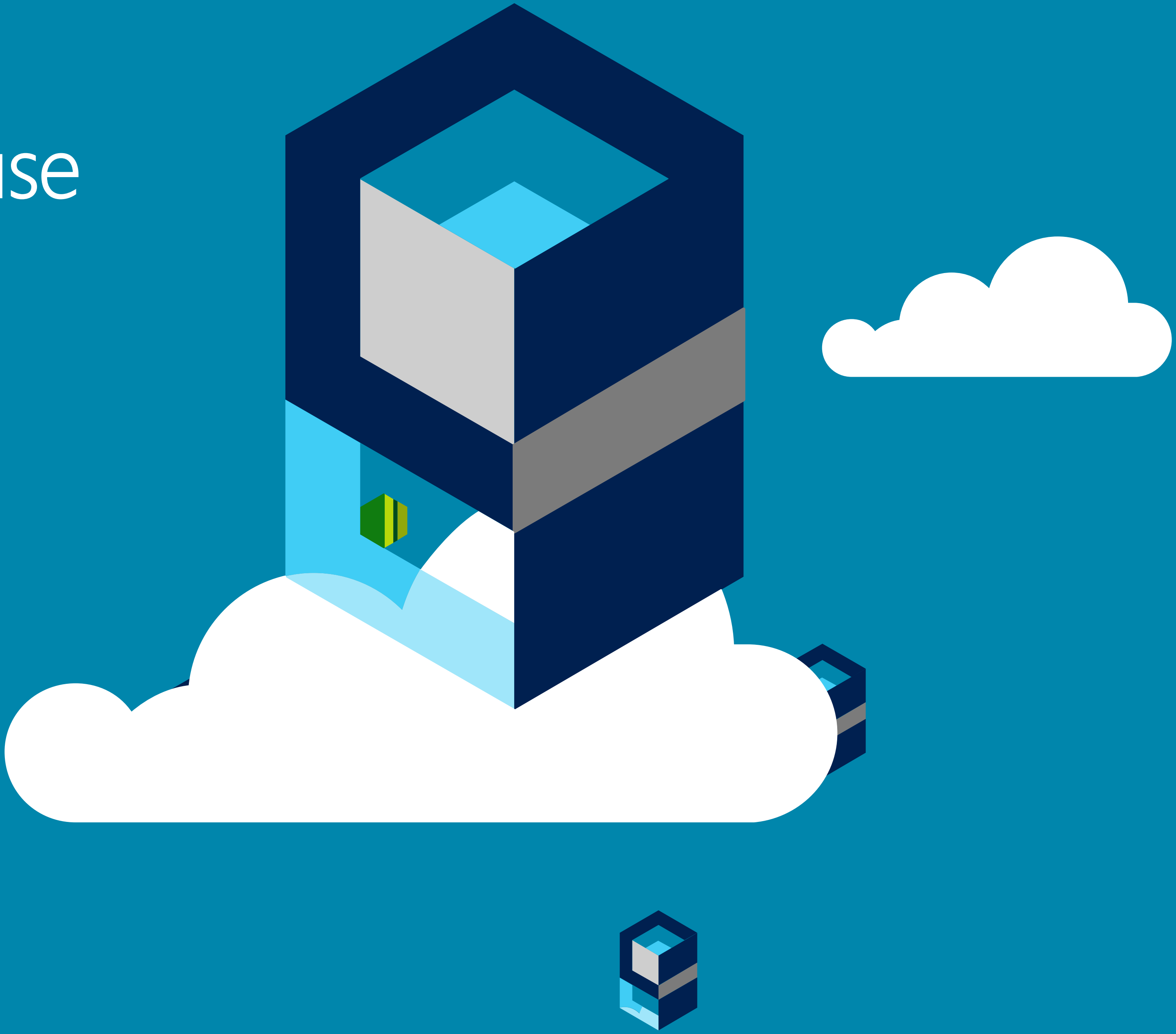
Microsoft

Azure Data Warehouse
In-A-Day



Keshav Pokkuluri
Casey Karst
Kal Yella

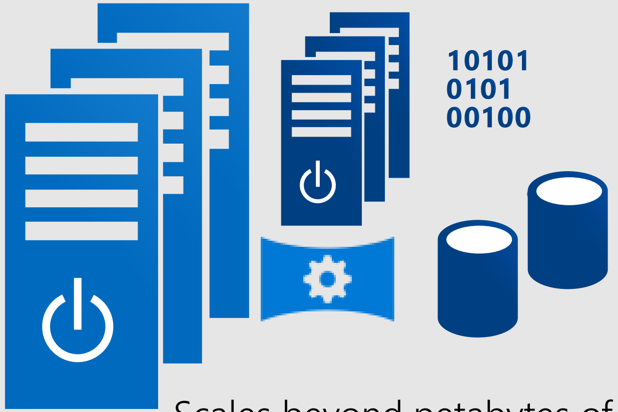
Agenda

8:45 AM to 9:00 AM	Welcome
9:00 AM to 9:45 AM	Datawarehouse Patterns & SQL DW Overview
9:45AM to 10:30 AM	SQL DW Gen2 New Features & Functionality
Break: 10:30 AM to 10:45 AM	
10:45 AM to 11:15 PM	SQLDW Loading Best Practices
11:15 AM to 12:00 PM	SQLDW Operational Best Practices
Lunch: 12:00 PM to 1:00 PM	SQLDW Roadmap Discussion
1:00 PM to 1:45 PM	SQL DW Compete Discussion
Break: 1:45 PM to 2:00 PM	
2:00 PM to 2:45 PM	Lab 1: Data loading scenarios and best practices
2:45 PM to 3:30 PM	Lab 2: Performance Tuning best practices
3:30 PM to 4:30 PM	Lab 3: Monitoring, Maintenance and Security
4:30 PM to 5:00 PM	Q&A and Wrap-up

Azure SQL DW Service

A relational **data warehouse-as-a-service**, fully managed by Microsoft.
Industries first **elastic** cloud data warehouse with proven SQL Server capabilities.
Support your **smallest to your largest** data storage needs.

Elastic scale & performance



Scales beyond petabytes of data

Cloud DW Scale-out Processing

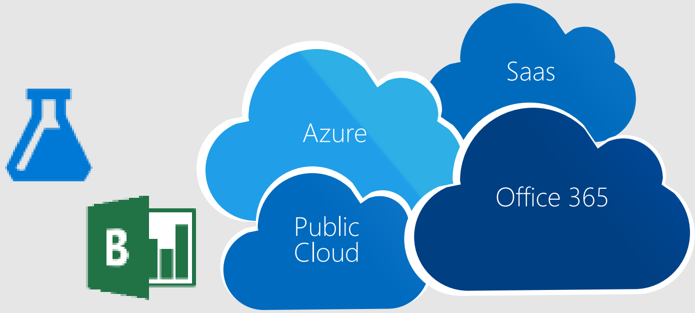
Instant-on compute scales in seconds

Query Relational / Non-Relational

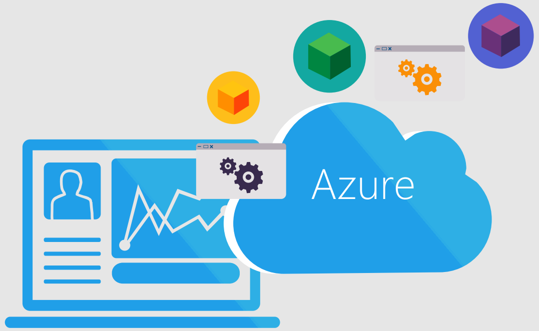
Powered by the Cloud

Get started in minutes

Integrated with Azure ML, PowerBI, ADB & ADF



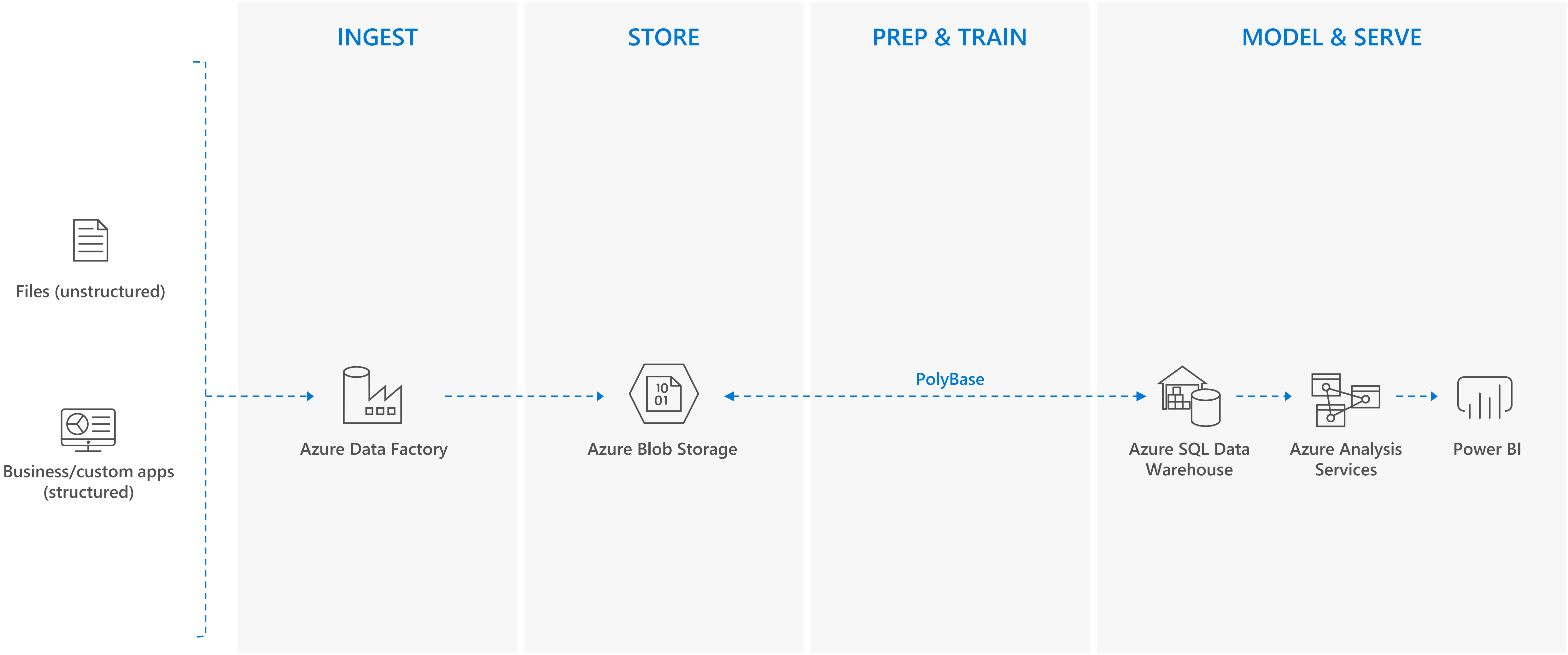
Market Leading Price & Performance



Simple billing compute & storage

Pay for what you need, when you need it with dynamic pause

CLOUD DATA WAREHOUSE



INGEST

STORE

PREP & TRAIN

MODEL & SERVE

Files (unstructured)

Business/custom apps (structured)

Azure Data Factory

Azure Blob Storage

PolyBase

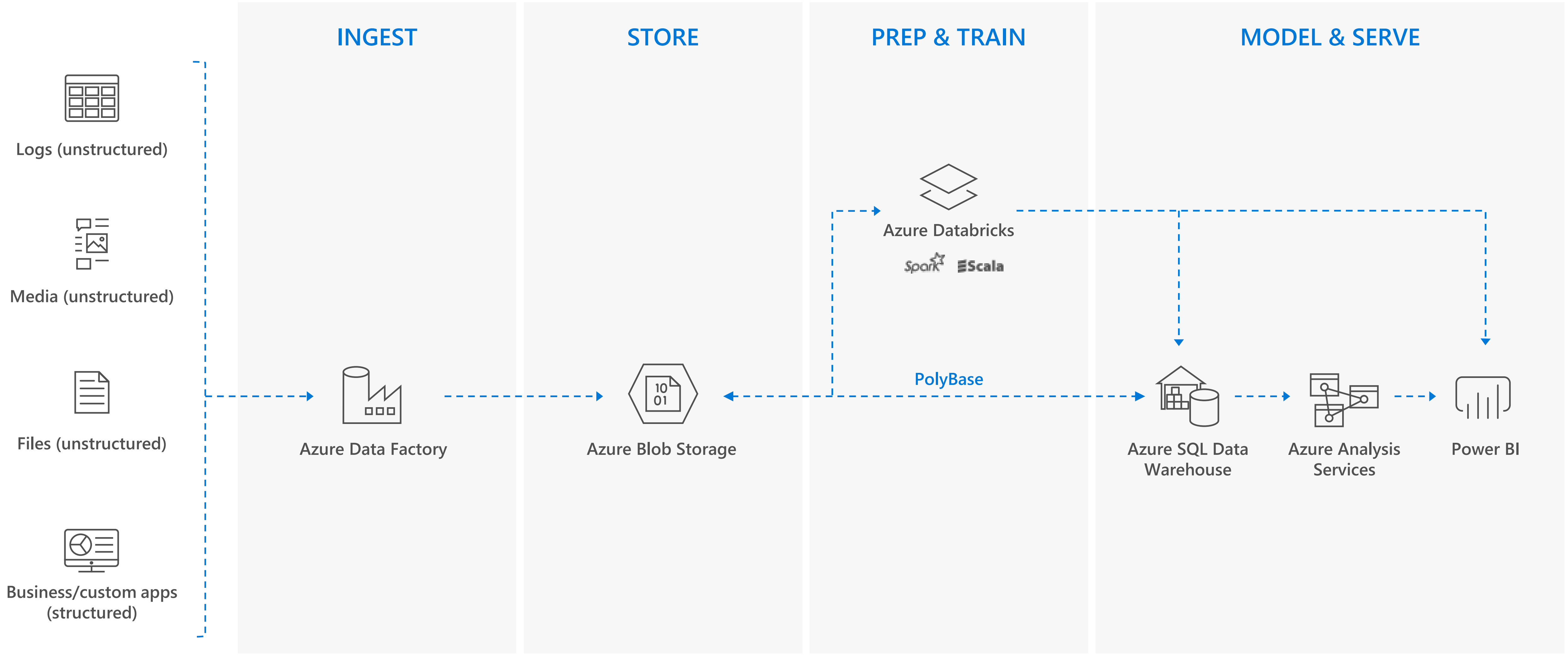
Azure SQL Data Warehouse

Azure Analysis Services

Power BI

Microsoft Azure also supports other Big Data services like Azure HDInsight and Azure Data Lake to allow customers to tailor the above architecture to meet their unique needs.

MODERN DATA WAREHOUSE



INGEST

STORE

PREP & TRAIN

MODEL & SERVE

Logs (unstructured)

Media (unstructured)

Files (unstructured)

Business/custom apps (structured)

Azure Data Factory

Azure Blob Storage

Azure Databricks

Spark Scala

PolyBase

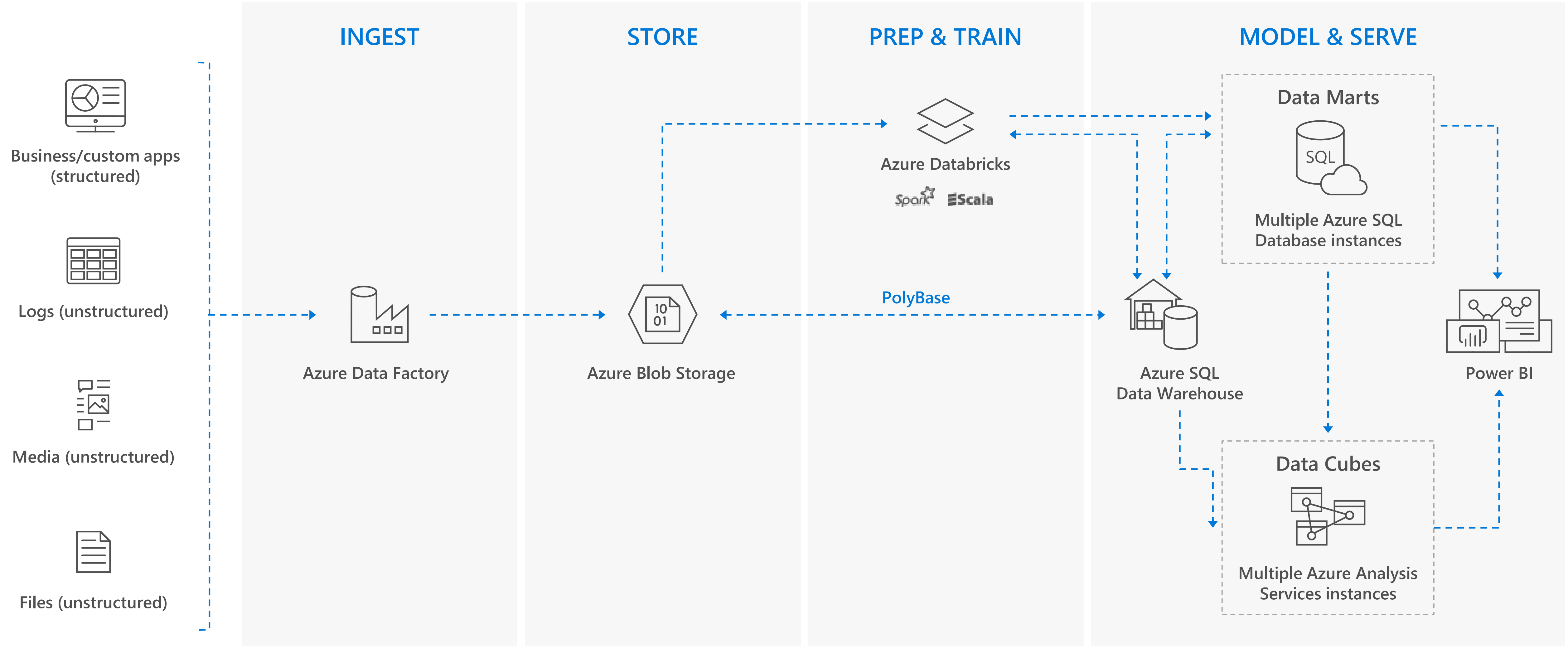
Azure SQL Data Warehouse

Azure Analysis Services

Power BI

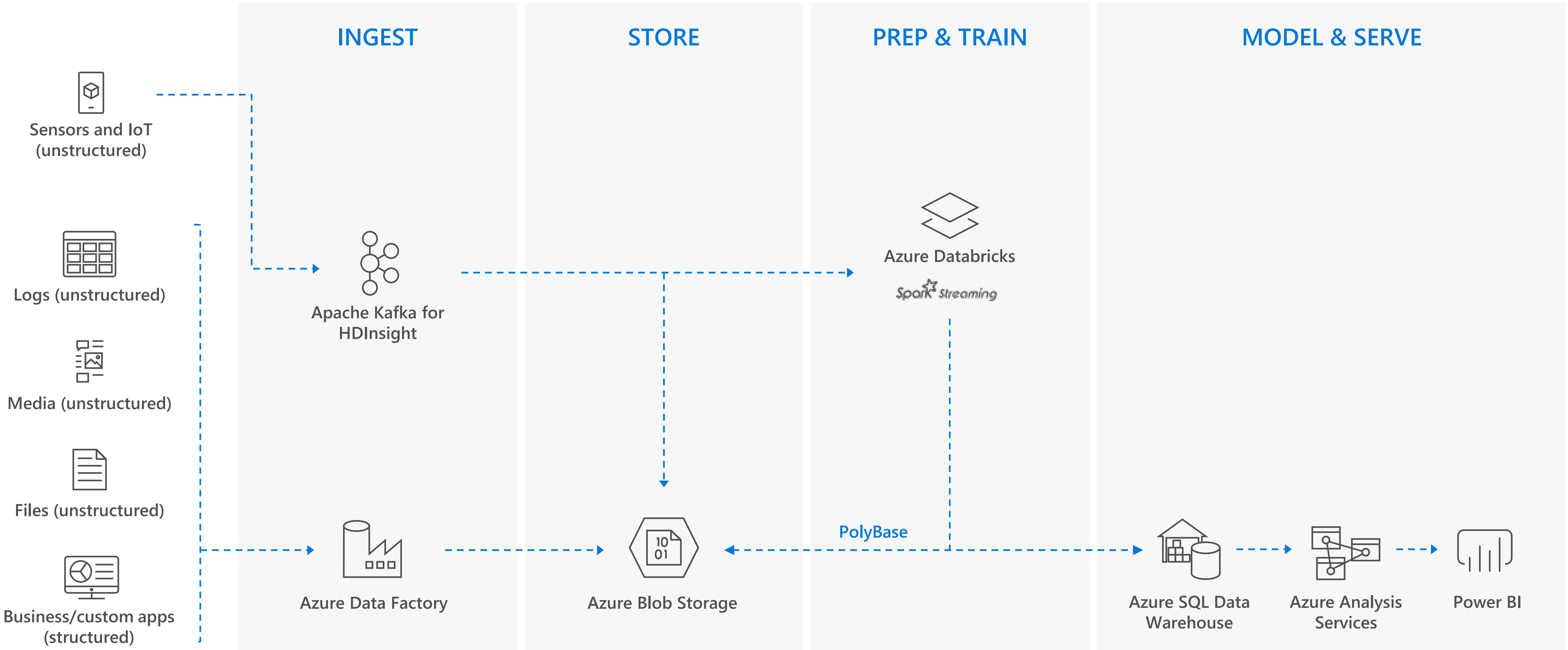
Microsoft Azure also supports other Big Data services like Azure HDInsight and Azure Data Lake to allow customers to tailor the above architecture to meet their unique needs.

HUB & SPOKE ARCHITECTURE FOR BI



Microsoft Azure supports other services like Azure HDInsight and Azure Data Lake in various layers to allow customers a truly customized solution.

REAL TIME ANALYTICS

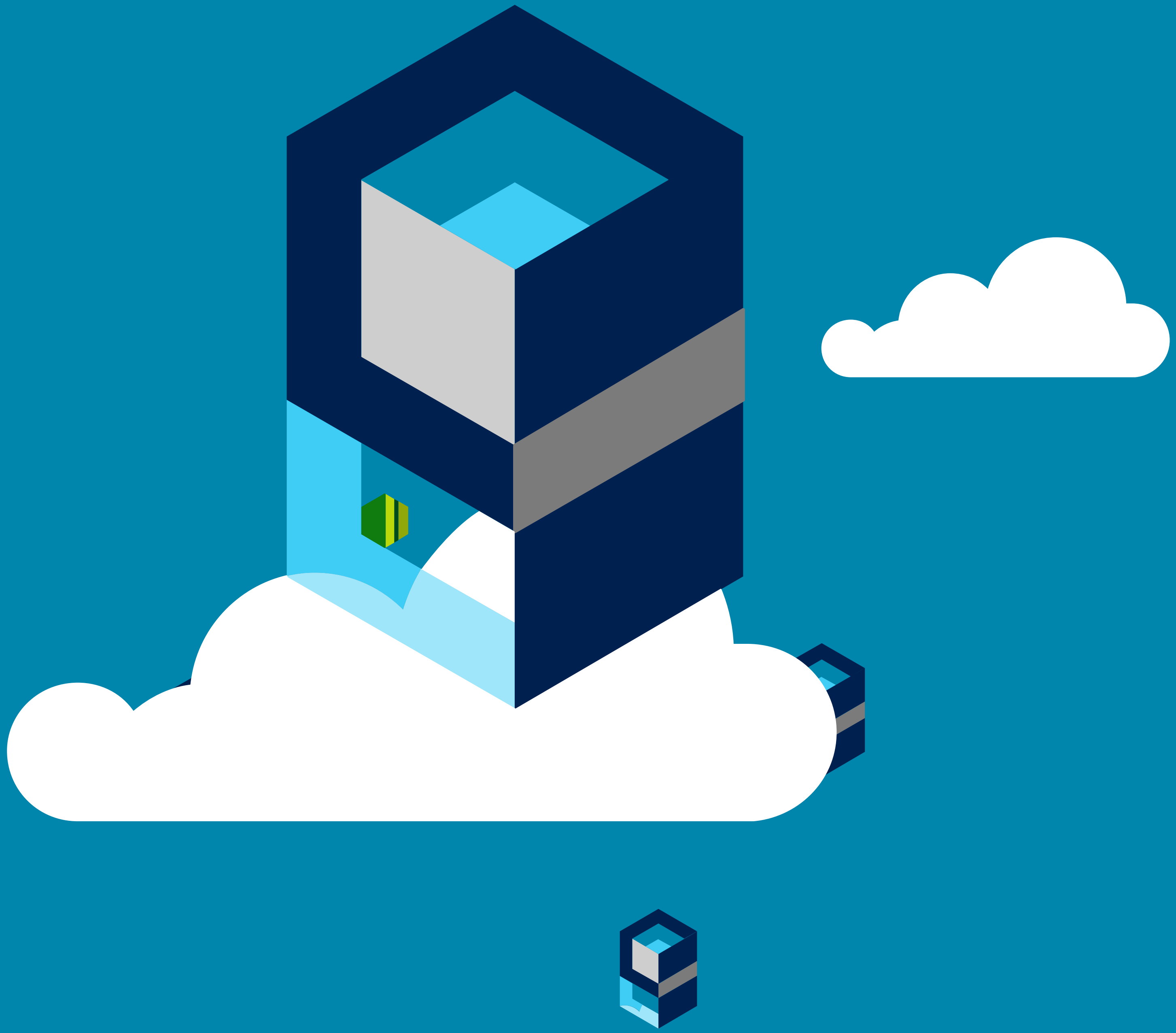


Microsoft Azure also supports other Big Data services like Azure IoT Hub, Azure Event Hubs, Azure Machine Learning and Azure Data Lake to allow customers to tailor the above architecture to meet their unique needs.

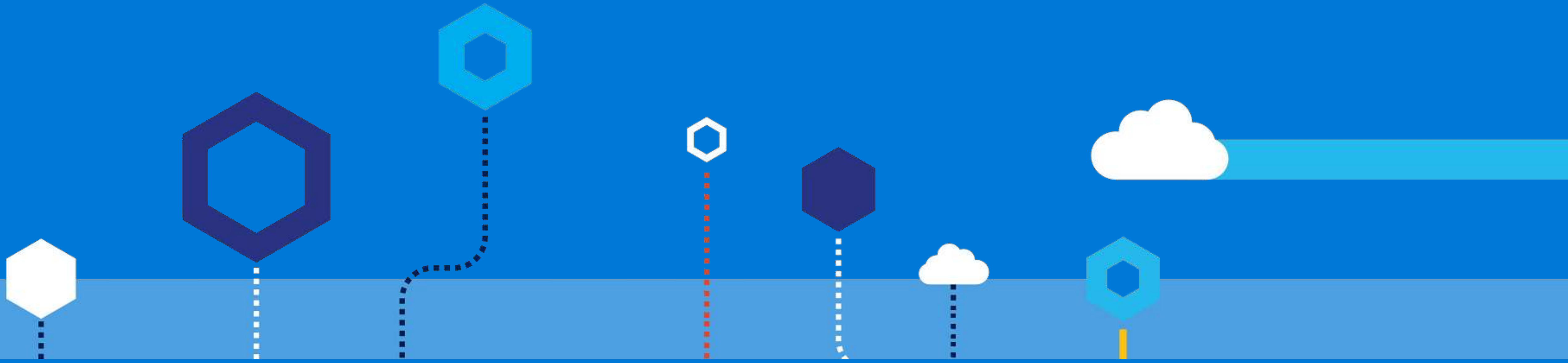
Microsoft

Azure SQL DW Overview

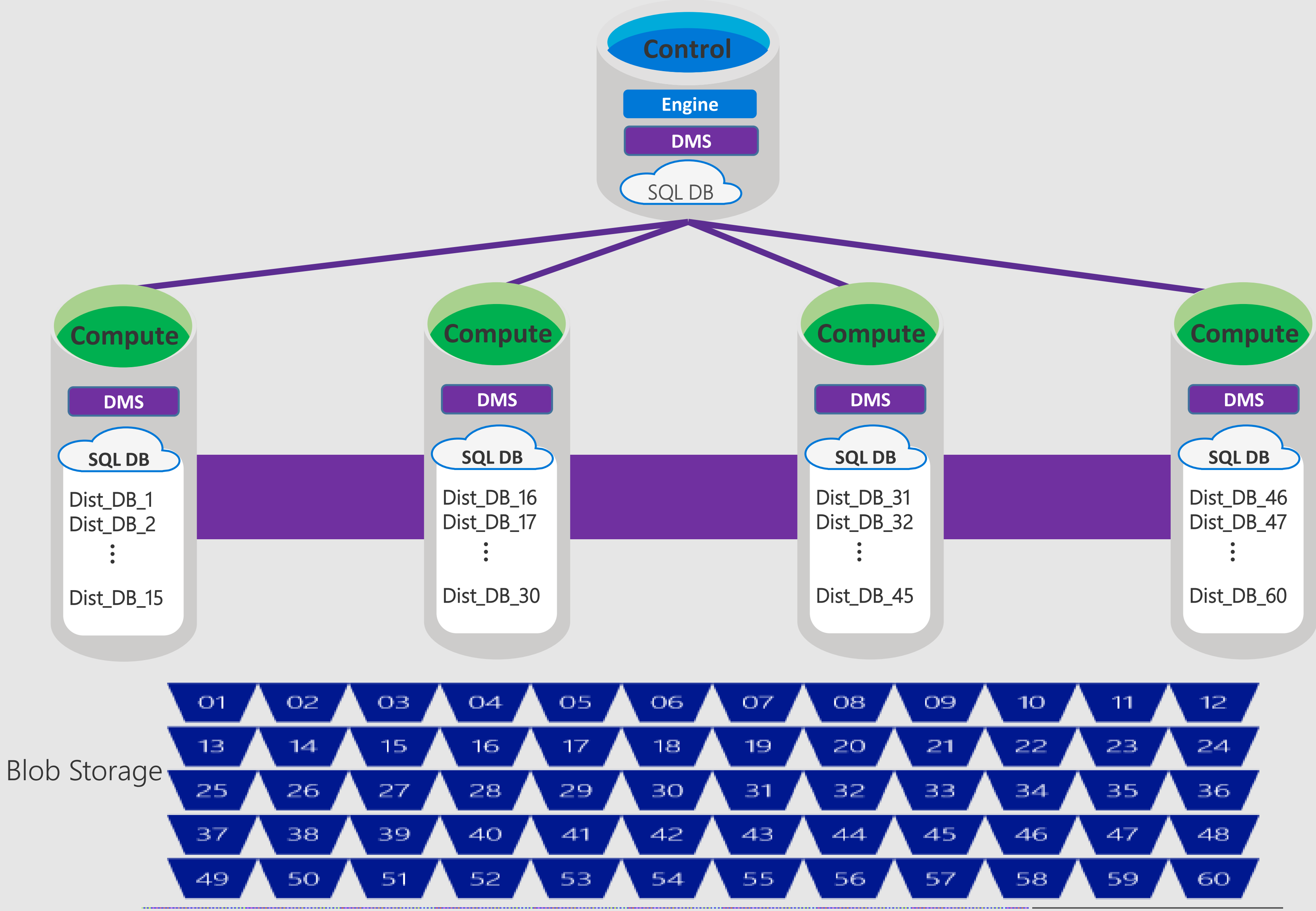
Azure SQL DW CSE



Under the hood - Architecture



SQL DW Fundamentals



Control
Connection and tool endpoint.
Coordinates compute activity.

Compute
Handles query processing, ability to scale up/down

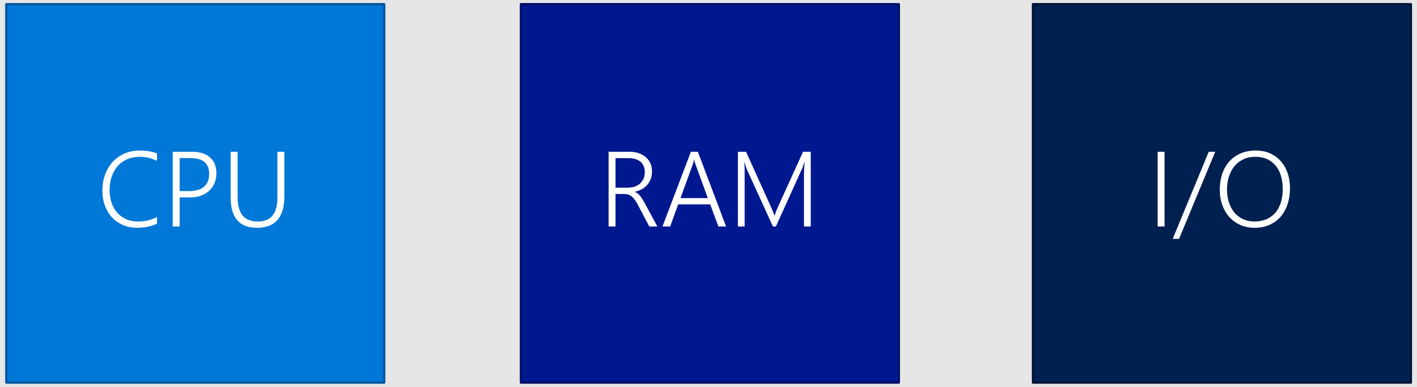
DMS: Data Movement Services
Coordinates data movement between nodes

Storage
Remote storage. Scales independently of compute

Data Warehouse Units

Normalized amount of compute

Converts to billing units i.e. what you pay



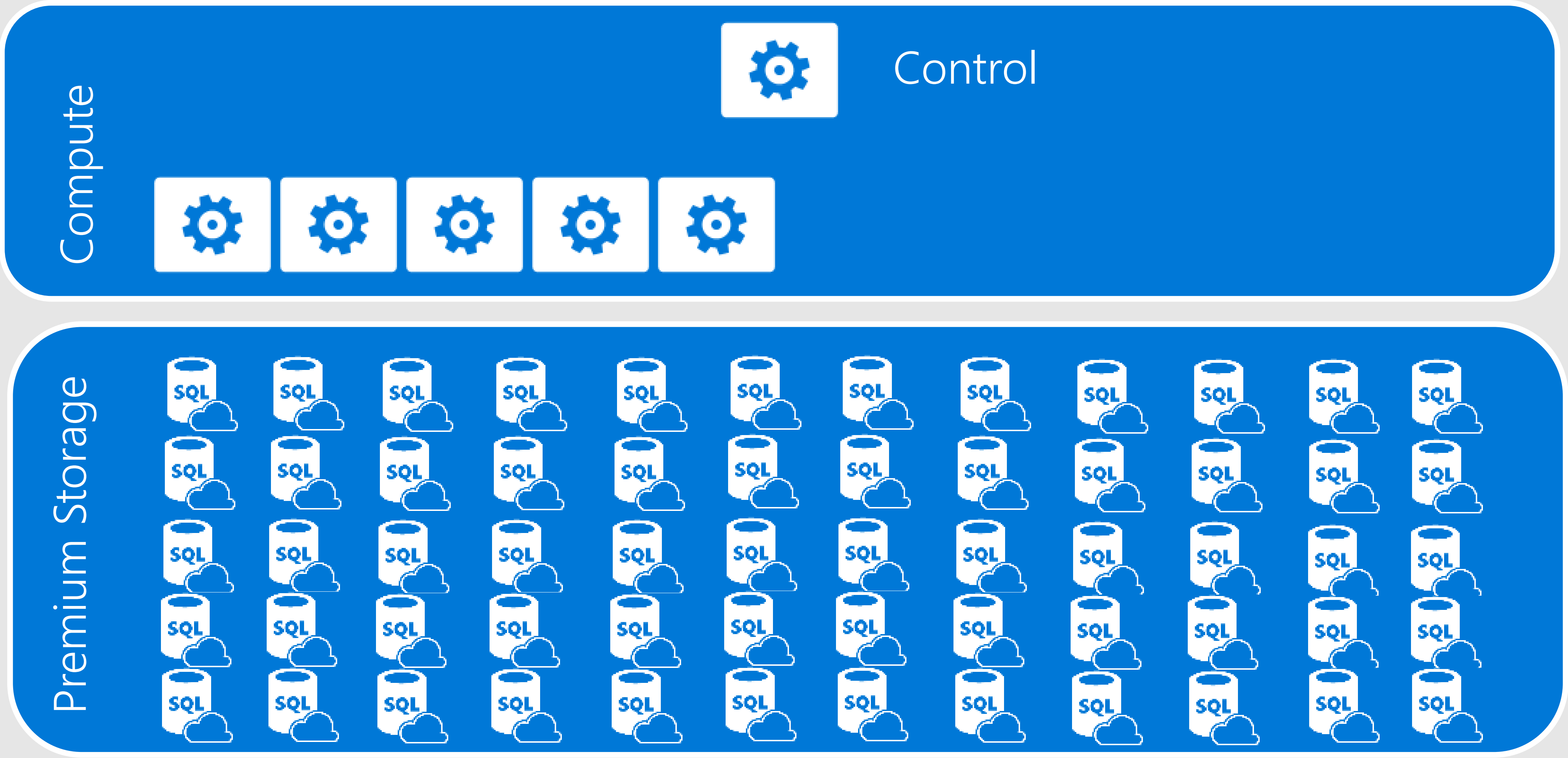
CPU

RAM

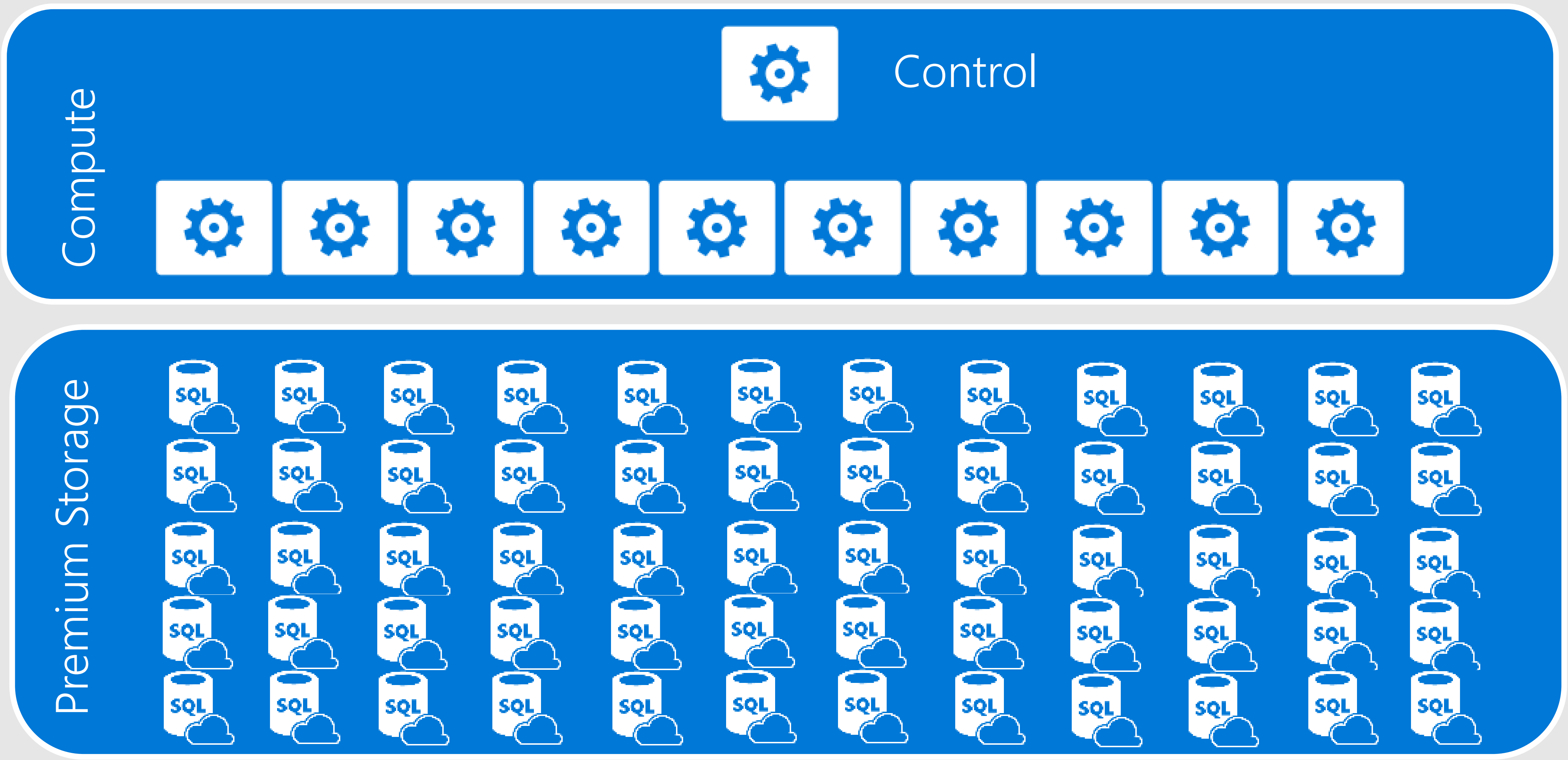
I/O

DWUc	Nodes	Dist/ Node
100	1	60
200	1	60
300	1	60
400	1	60
500	1	60
1000	2	30
1500	3	20
2000	4	15
2500	5	12
3000	6	10
5000	10	6
…		
30000	60	1

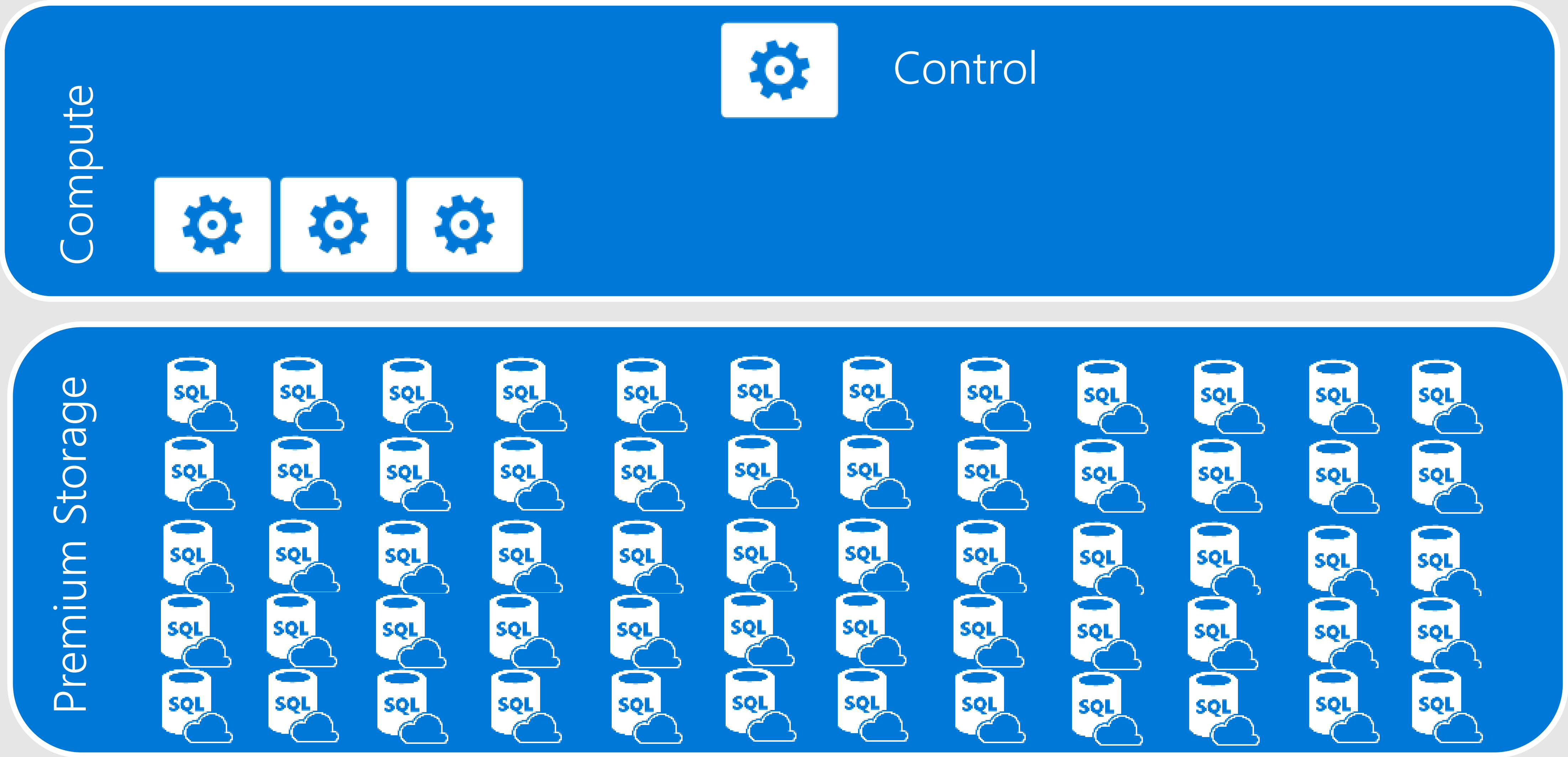
Separate compute from storage



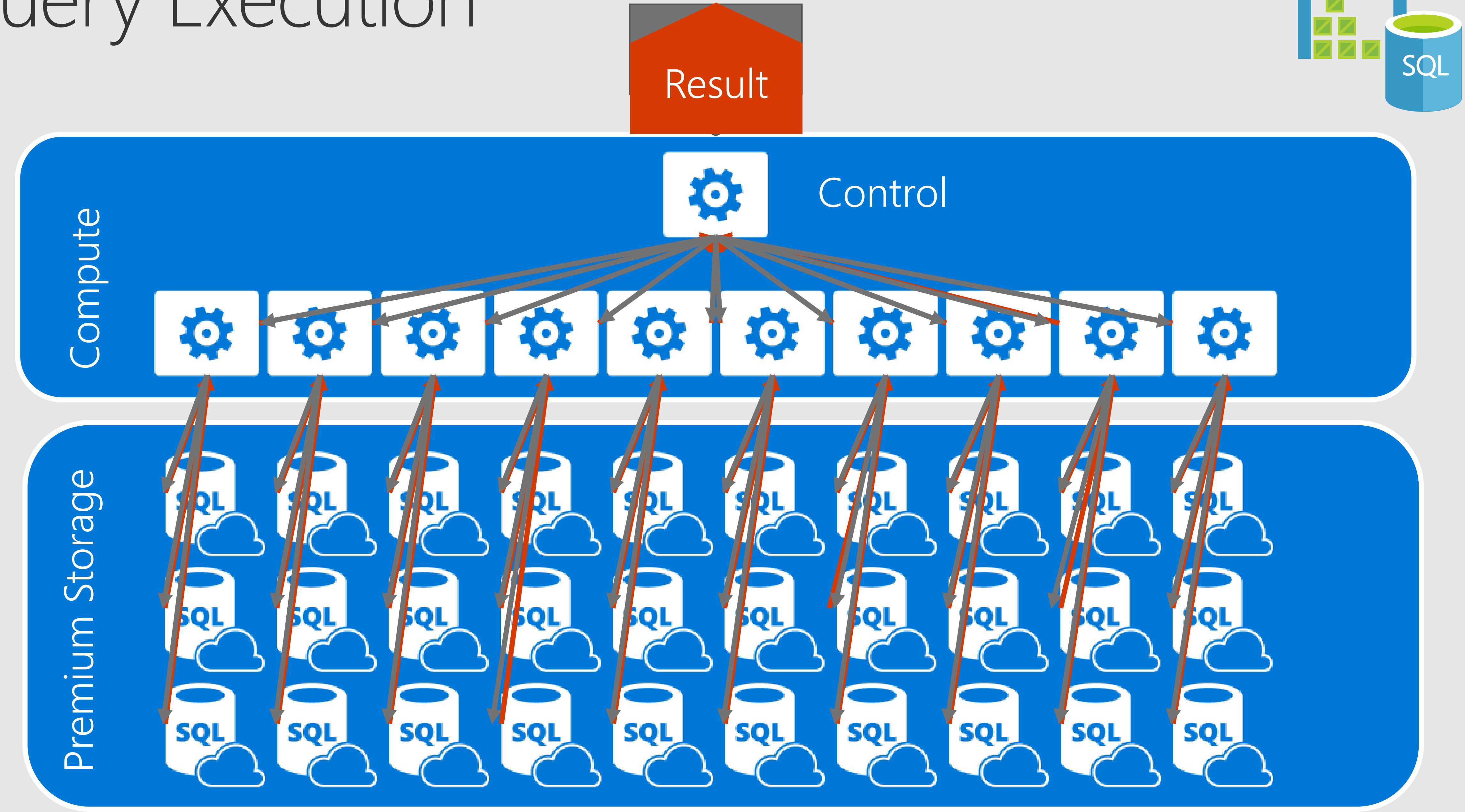
Independently scale compute



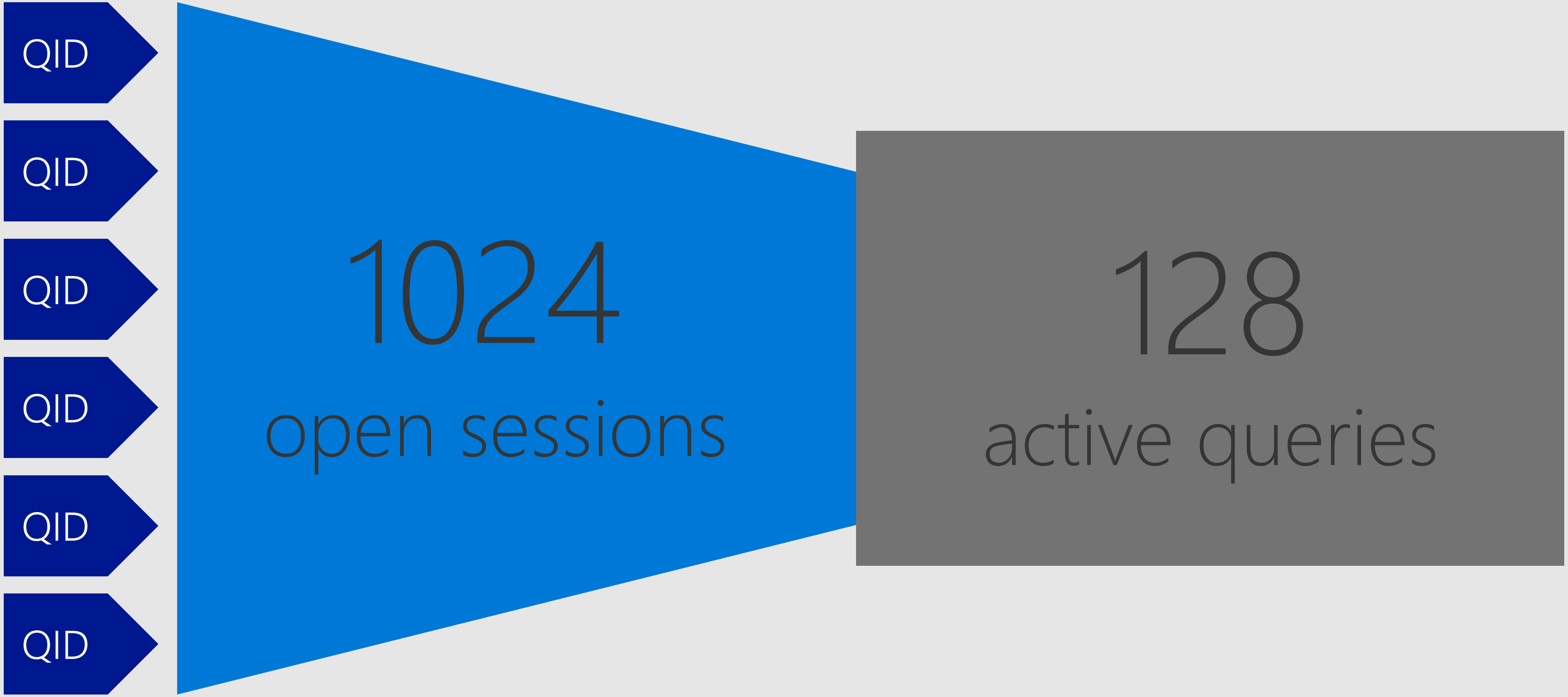
Pause and resume workload



Query Execution



Concurrent queries



QID

QID

QID

QID

QID

QID

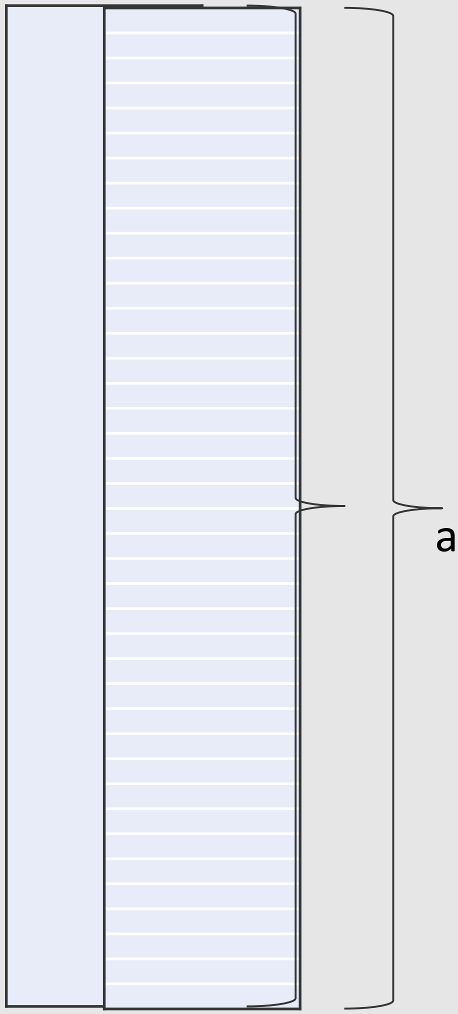
1024

open sessions

128

active queries

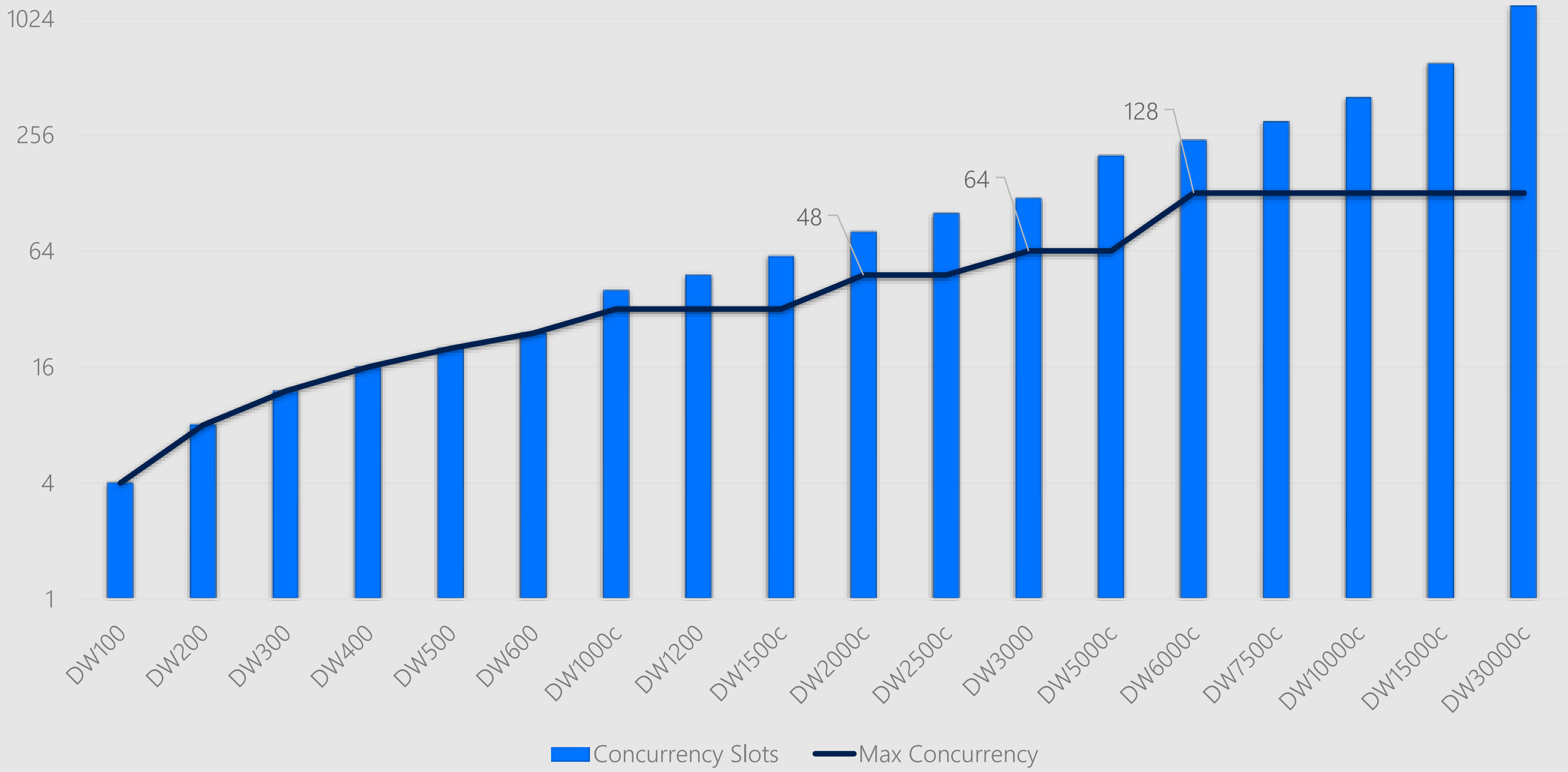
Concurrency slots



Total system
available memory

Concurrency Slots
20 per DW500c: 250MB/slot
4 per DW100c: 100MB/slot

Concurrent Query and Slots



Resource classes

Dynamic

Increases resource consumption as you scale

No increase in concurrency as you scale

Static

Maintain resource consumption as you scale

Increase concurrent queries as you scale

Consume Slots

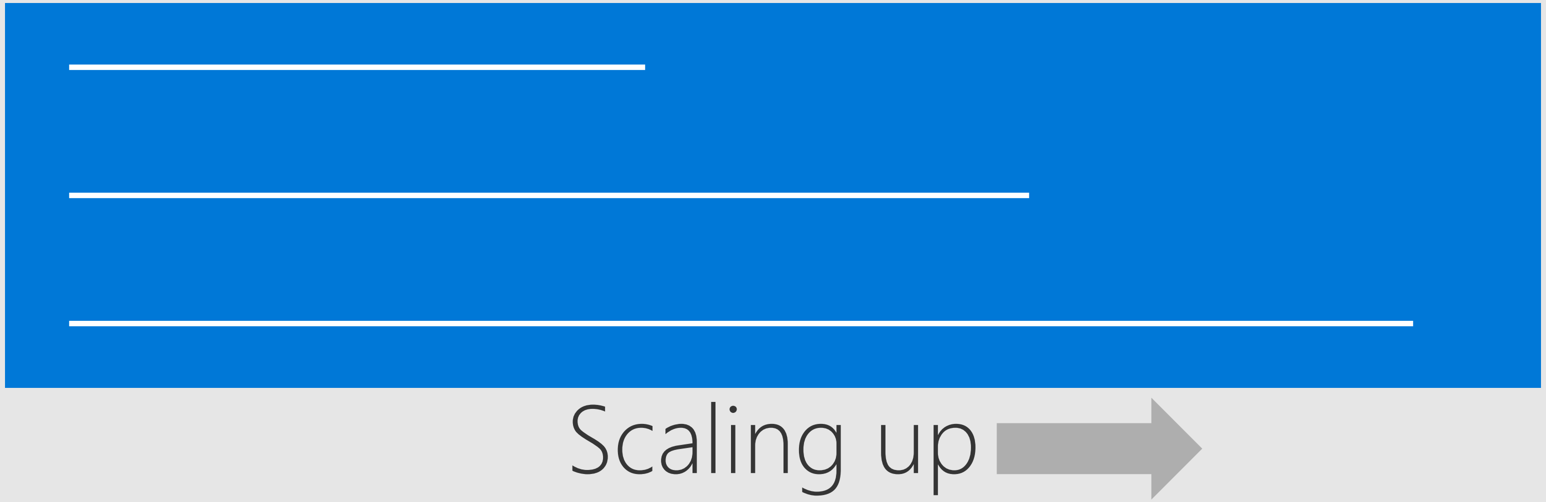
Increase memory
Isolate resources

Resource Classes – Dynamic

Allocates variable amounts of memory depending on the scale of the DW instance.

✓ Beneficial for variable sized workloads that scale to meet demand.

There is no increase in concurrency with scaling.



Scaling up

Dynamic Resource Classes

With Gen2, dynamic resource pools were introduced with a 3-10-22-70 model for resource allocations.

Resource Class	Percent Resources	Concurrency
SmallRc	3%	32
MediumRc	10%	10
LargeRc	22%	4
XLargeRc	70%	1

Resource Classes – Static

Allocates a fixed amount of memory regardless of the scale level.

✓ Essential for high query concurrency workloads.

Queries may run the same regardless of the service level.

Gen 2 Concurrency – Static RC

Service Level	Maximum concurrent queries	Concurrency slots available	staticrc10	staticrc20	staticrc30	staticrc40	staticrc50	staticrc60	staticrc70	staticrc80
DW100c	4	4	1	2	4	4	4	4	4	4
DW200c	8	8	1	2	4	8	8	8	8	8
DW300c	12	12	1	2	4	8	8	8	8	8
DW400c	16	16	1	2	4	8	16	16	16	16
DW500c	20	20	1	2	4	8	16	16	16	16
DW1000c	32	40	1	2	4	8	16	32	32	32
DW1500c	32	60	1	2	4	8	16	32	32	32
DW2000c	48	80	1	2	4	8	16	32	64	64
DW2500c	48	100	1	2	4	8	16	32	64	64
DW3000c	64	120	1	2	4	8	16	32	64	64
DW5000c	64	200	1	2	4	8	16	32	64	128
DW6000c	128	240	1	2	4	8	16	32	64	128
DW7500c	128	300	1	2	4	8	16	32	64	128
DW10000c	128	400	1	2	4	8	16	32	64	128
DW15000c	128	600	1	2	4	8	16	32	64	128
DW30000c	128	1200	1	2	4	8	16	32	64	128

Data Distribution

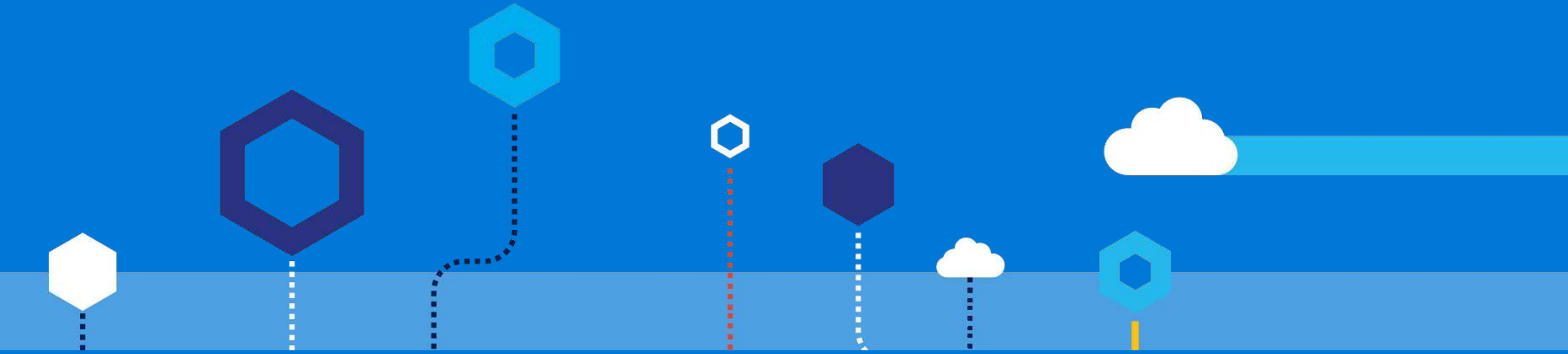


Table Distribution Options

Hash Distributed

Data divided across nodes based on hashing algorithm

Same value will always hash to same distribution

Single column only

Check for Data Skew, NULLS, -1

Round Robin
(Default)

Data distributed evenly across nodes

Easy place to start, don’t need to know anything about the data

Simplicity at a cost

Will incur more data movement at query time

Replicated

Data repeated on every node

Simplifies many query plans and reduces data movement

Best with joining hash table

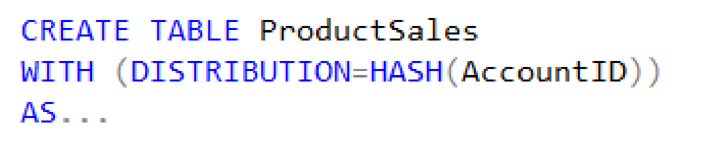
Consumes more space
Joining two Replicated Table runs
on one node

Creating tables

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]          int          NOT NULL
,   [DateKey]                 datetime     NOT NULL
,   [StoreKey]                int          NOT NULL
,   [ProductKey]              int          NOT NULL
,   [PromotionKey]            int          NOT NULL
,   [CurrencyKey]             int          NOT NULL
,   [CustomerKey]             int          NOT NULL
,   [SalesOrderNumber]        nvarchar(20) NOT NULL
,   [SalesOrderLineNumber]    int              NULL
,   [SalesQuantity]           int          NOT NULL
,   [SalesAmount]             money        NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = ROUND_ROBIN
)
;
```

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]          int          NOT NULL
,   [DateKey]                 datetime     NOT NULL
,   [StoreKey]                int          NOT NULL
,   [ProductKey]              int          NOT NULL
,   [PromotionKey]            int          NOT NULL
,   [CurrencyKey]             int          NOT NULL
,   [CustomerKey]             int          NOT NULL
,   [SalesOrderNumber]        nvarchar(20) NOT NULL
,   [SalesOrderLineNumber]    int              NULL
,   [SalesQuantity]           int          NOT NULL
,   [SalesAmount]             money        NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([ProductKey])
)
;
```

Hash Distributed



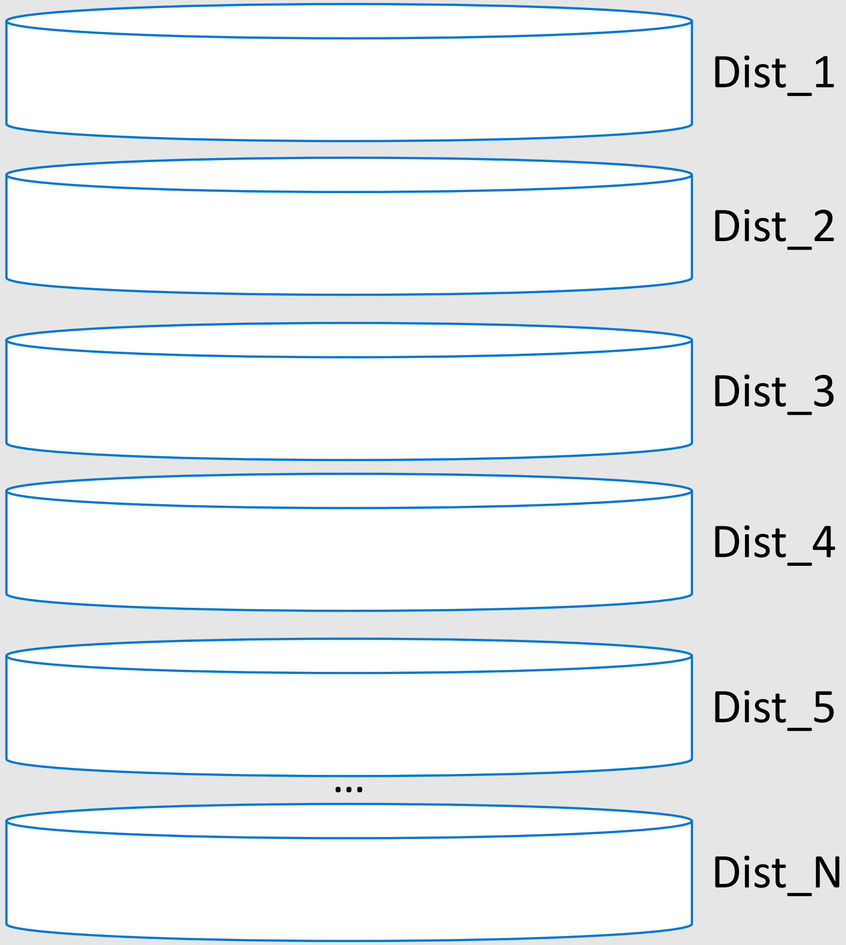
```
CREATE TABLE ProductSales
WITH (DISTRIBUTION=HASH(AccountID))
AS...
```

ProductSales – Raw Data

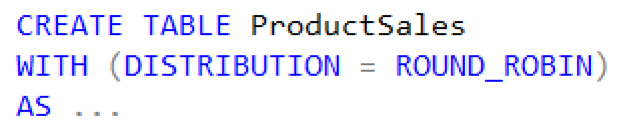
AccountID	SalesAmt	…
47	$1,234.36	…
36	$2,345.47	…
14	$3,456.58	…
25	$4,567.69	…
48	$5,678.70	…
37	$6,789.81	…
…	…	…

Hash(47)

Hash(36)



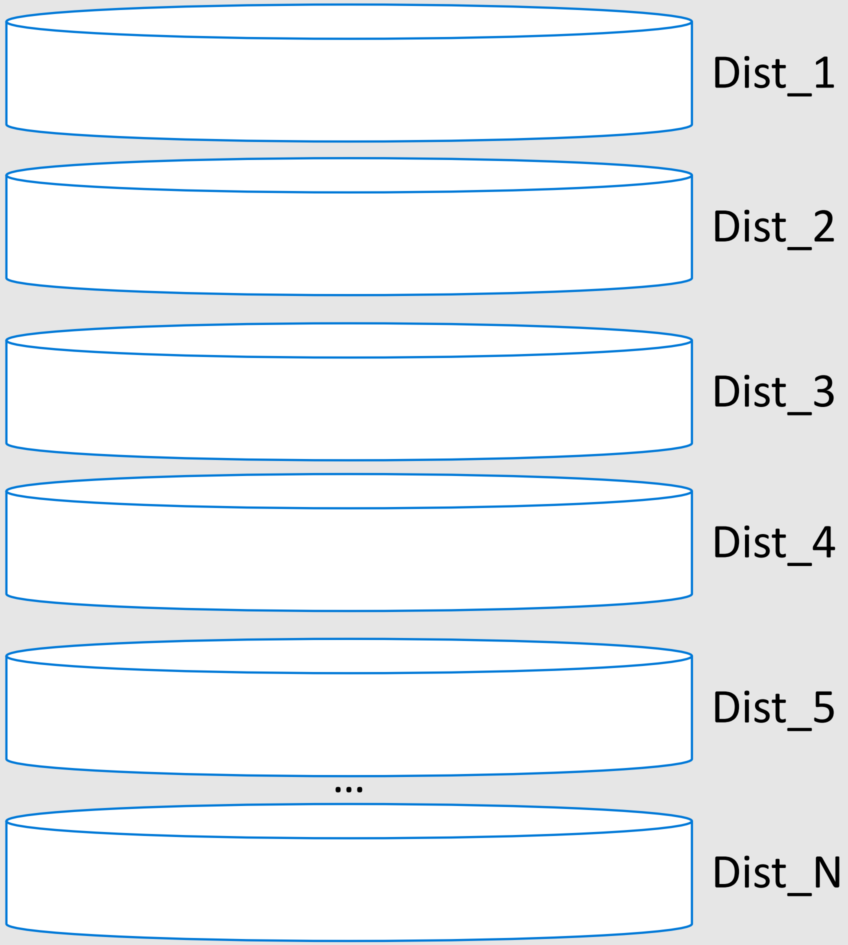
Round Robin Distributed



```
CREATE TABLE ProductSales
WITH (DISTRIBUTION = ROUND_ROBIN)
AS ...
```

ProductSales – Raw Data

AccountID	SalesAmt	…
47	$1,234.36	…
36	$2,345.47	…
14	$3,456.58	…
25	$4,567.69	…
48	$5,678.70	…
37	$6,789.81	…
42	$1,632.25	…
42	$4,453.21	…
52	$7,892.81	…
91	$9,549.64	…
66	$2,498.14	…
23	$3,145.99	…



T-SQL: Create a Replicated table

```
CREATE TABLE dbo.DimCustomer
(   CustomerKey          int                NOT NULL
,   GeographyKey         int                NULL
,   CustomerAlternateKey nvarchar(15)       NOT NULL
,   Title                nvarchar(8)        NULL
,   FirstName            nvarchar(50)       NULL
,   LastName             nvarchar(50)       NULL
,   BirthDate            date               NULL
,   Gender               nvarchar(1)        NULL
,   EmailAddress         nvarchar(50)       NULL
,   YearlyIncome         money              NULL
,   DateFirstPurchase    date               NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = REPLICATED
)
```

Indexing

Indexing Choices

- Row Store : Clustered Index
- Column Store : Clustered Columnstore Index
- Heap : No index
- Non-Clustered Index

Indexing tables

```sql
CREATE TABLE [dbo].[DimStore]
(
    [StoreKey]          int             NOT NULL
,   [GeographyKey]      int             NOT NULL
,   [StoreName]         nvarchar(100)   NOT NULL
,   [StoreType]         nvarchar(15)        NULL
,   [StoreDescription]  nvarchar(300)   NOT NULL
,   [Status]            nvarchar(20)    NOT NULL
,   [OpenDate]          datetime        NOT NULL
,   [CloseDate]         datetime            NULL
,   [ETLLoadID]         int                 NULL
,   [LoadDate]          datetime            NULL
,   [UpdateDate]        datetime            NULL
)
WITH
(   CLUSTERED INDEX([StoreKey])
,   DISTRIBUTION = ROUND_ROBIN
)
;
```

Row

```sql
CREATE TABLE [dbo].[FactOnlineSales]
(
    [OnlineSalesKey]        int           NOT NULL
,   [DateKey]               datetime      NOT NULL
,   [StoreKey]              int           NOT NULL
,   [ProductKey]            int           NOT NULL
,   [PromotionKey]          int           NOT NULL
,   [CurrencyKey]           int           NOT NULL
,   [CustomerKey]           int           NOT NULL
,   [SalesOrderNumber]      nvarchar(20)  NOT NULL
,   [SalesOrderLineNumber]  int               NULL
,   [SalesQuantity]         int           NOT NULL
,   [SalesAmount]           money         NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([ProductKey])
)
;
```

Column

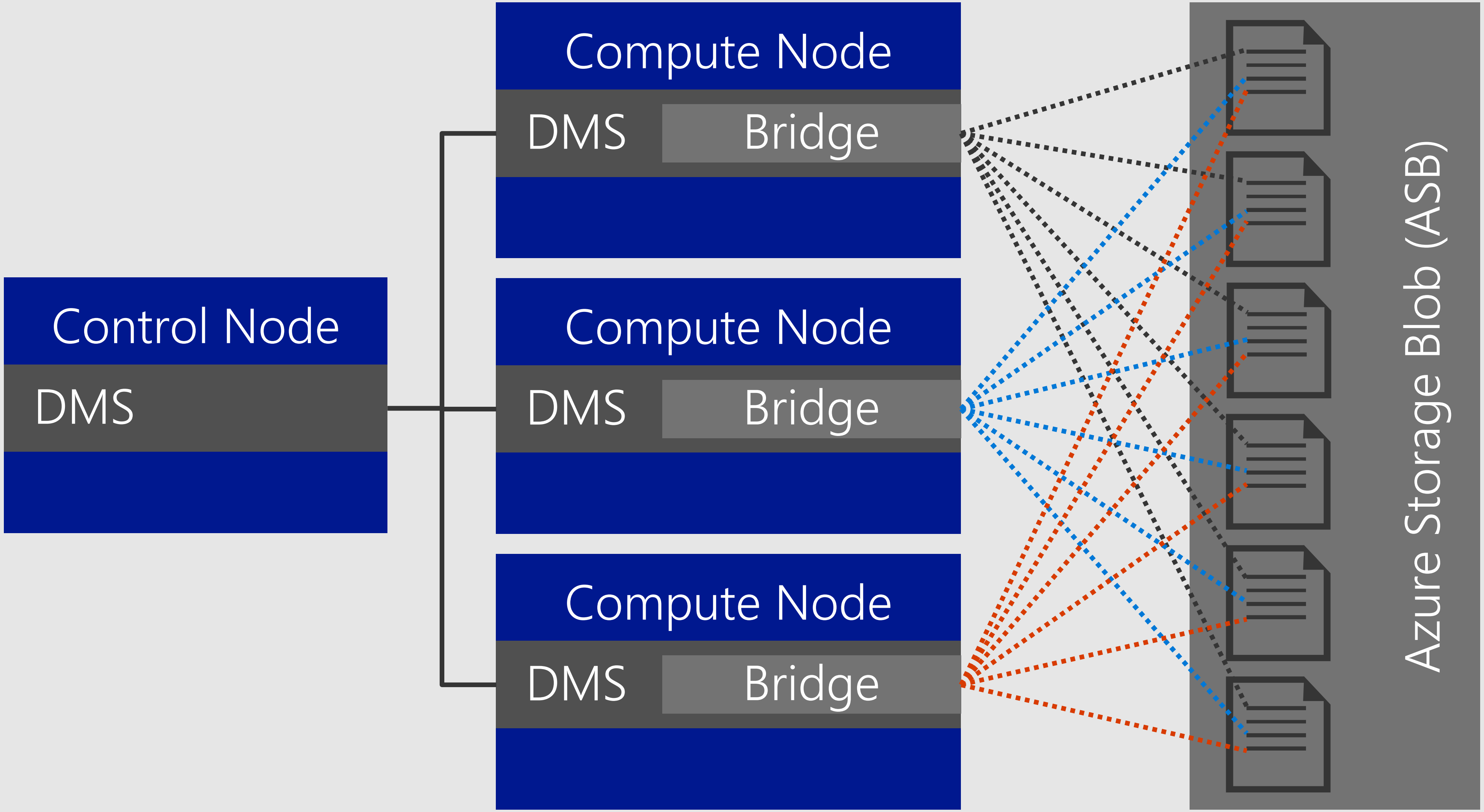
Distribution

Clustered Columnstore & Partitioning

```
CREATE TABLE [dbo].[FactInternetSales]
(
[ProductKey] int NOT NULL ,
[OrderDateKey] int NOT NULL ,
[CustomerKey] int NOT NULL ,
[PromotionKey] int NOT NULL ,
[SalesOrderNumber] nvarchar(20) NOT NULL ,
[OrderQuantity] smallint NOT NULL ,
[UnitPrice] money NOT NULL ,
[SalesAmount] money NOT NULL
)
WITH ( CLUSTERED COLUMNSTORE INDEX ,
       DISTRIBUTION = HASH([ProductKey]) ,
       PARTITION ( [OrderDateKey] RANGE RIGHT FOR
       VALUES (20000101,20010101,20020101,20030101,20040101,20050101)
              )
       )
 ;
```

Polybase

Polybase parallel load from Azure Storage



Additional Resources

Azure site: http://aka.ms/sqldw

SQL DW suggestions: http://aka.ms/sql-dw-feedback

Stack Overflow – Tag: azure-sqldw

Twitter: @AzureSQLDW

Nominate: http://aka.ms/engage_dw_cse

Microsoft

Q&A

Microsoft

Modernizing Your Data Warehouse