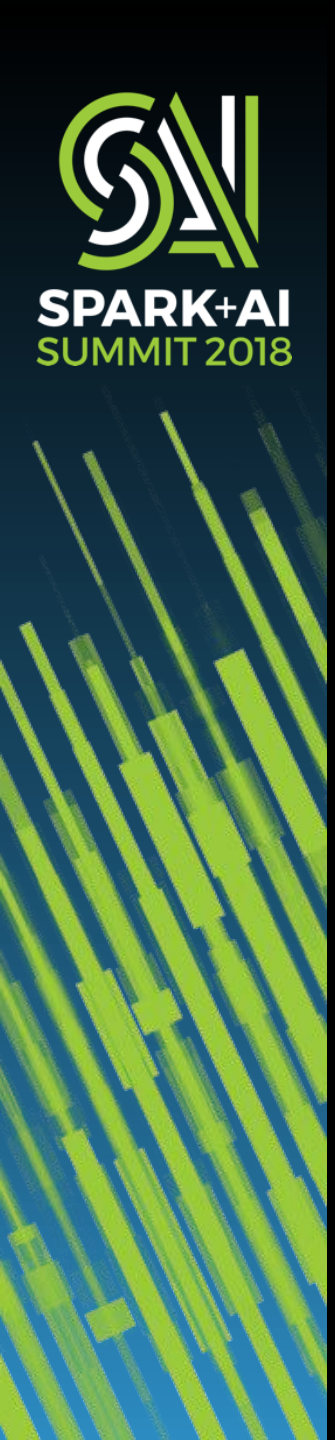


Azure Databricks Customer Experiences and Lessons

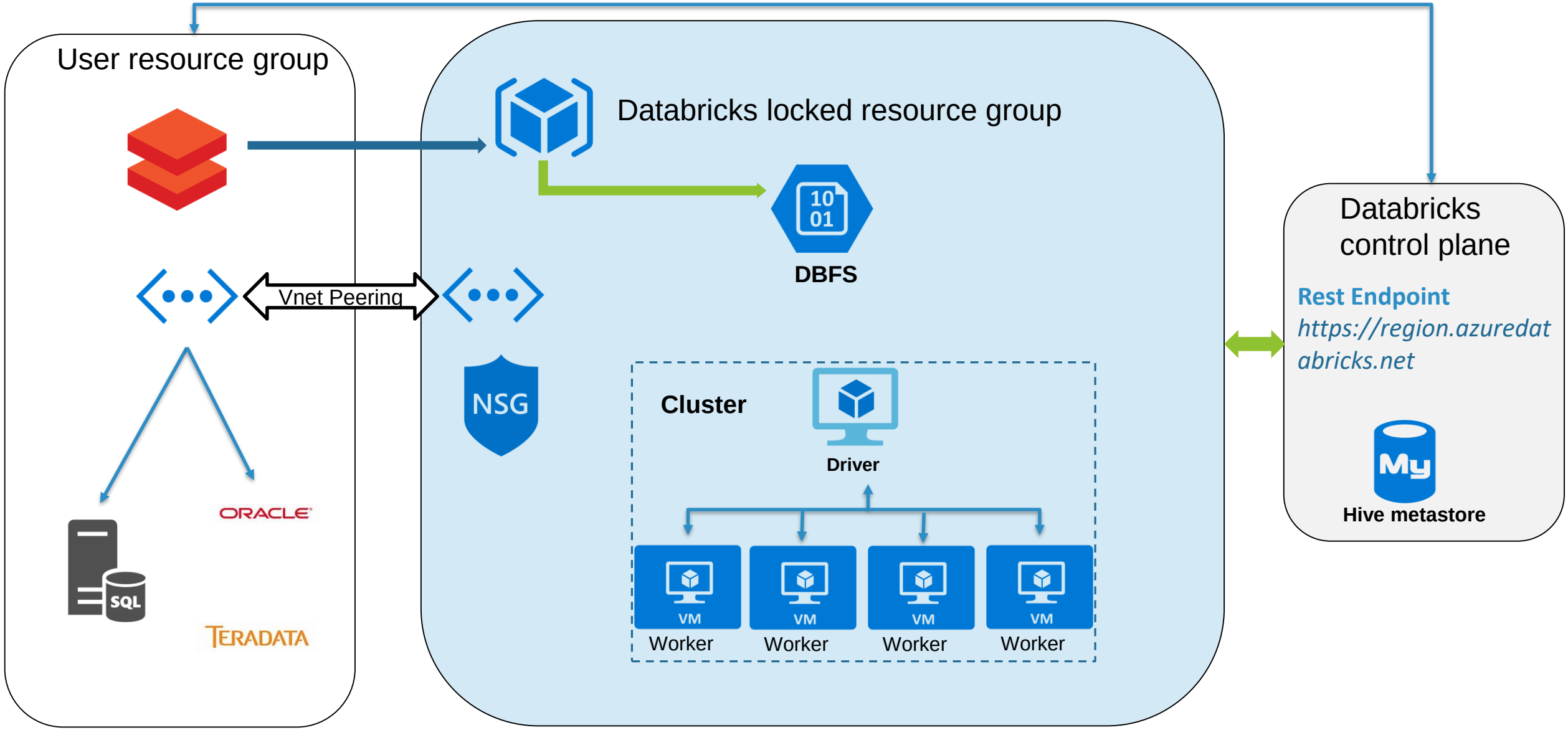
Denzil Ribeiro & Madhu Ganta

Microsoft

Objectives

- Understand customer deployment of Azure Databricks
- Understand customer integration requirements on Azure platform
- Best practices on Azure Databricks

Azure Databricks Deployment



What VM should I pick for my workload?

VM SKU Impact

- Different VM SKU can be chosen for driver v/s workers
- Cores (Impacts parallelism)
- Memory (Impacts JVM heap)
- Local SSD Size & IOPS Limits (Impacts DBIO cache & spills)
- Attached Disk Throughput Limits (Impacts DBIO cache & spills)
- Network Throughput Limits(Impacts shuffle)

SKU	vCPU	Memory (GiB)	Local SSD Size (GiB)	Max Cached Local SSD Throughput (IOPS / MBps)	Max uncached attached disk Throughput (IOPS / MBps)	Max NICs / network bandwidth (Mbps)
Standard_DS3_v2	4	14	28	16,000 / 128	12,800 / 192	4 / 3000
Standard_DS14	16	112	224	64,000 / 512	51,200 / 512	8 / 8000
Standard_L16s	16	128	2,807	80,000 / 800	20,000 / 500	8 / 16,000

Where can I store my data?

DBFS

- Built-in Distributed File system tied to workspace
- Layer on top of Azure storage - default setup gives no “control” outside of databricks workspace.
- Other file systems can be mounted on to DBFS

Azure Blob Storage

- Managed azure service providing highly redundant scalable, secure storage
- Data can be accessed via storage Keys or SAS tokens
- V2 storage accounts have higher limits

Azure Data Lake Store

- Databricks can access data using Azure Active Directory Authentication and Authorization
- Allows ACLs on data secured via Azure Service Principal Names (SPNs)
- Region availability less than Azure Blob storage

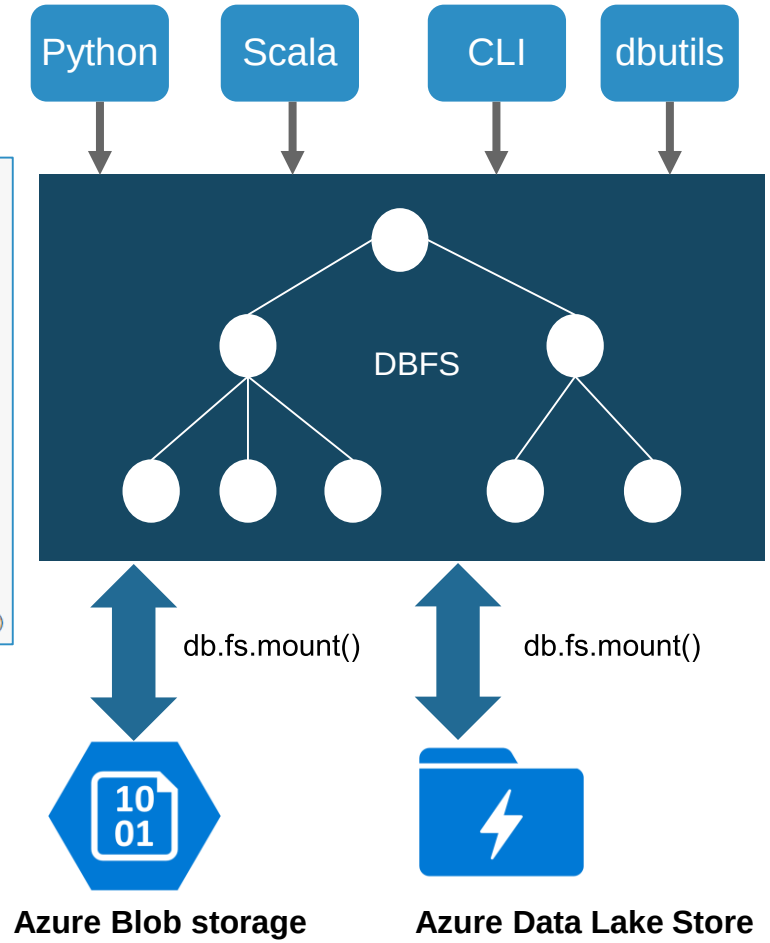
How do I mount external storage?

Storage can be mounted on DBFS so that users can directly access data without specifying the storage credentials

Mount points are at the workspace level, not cluster level

```
// Azure Data Lake - Mount once per workspace
val configs = Map(
  "dfs.adls.oauth2.access.token.provider.type" -> "ClientCredential",
  "dfs.adls.oauth2.client.id" -> "{YOUR SERVICe CLIENT ID}",
  "dfs.adls.oauth2.credential" -> "{YOUR SERVICE CREDENTIALS}",
  "dfs.adls.oauth2.refresh.url" -> "https://login.windows.net/{YOUR DIRECTORY ID}/oauth2/token")
dbutils.fs.mount(
  source = "adl://{YOUR DATA LAKE ACCOUNT}.azuredatalakestore.net/",
  mountPoint = "/mnt/adl",
  extraConfigs = configs)

//Azure Blob Storage - Mount once per workspace
dbutils.fs.mount(
  source = "wasbs://{YOUR STORAGE ACCOUNT}.blob.core.windows.net/",
  mountPoint = "/mnt/wasb",
  extraConfigs = Map("fs.azure.account.key.{YOUR STORAGE ACCOUNT}.blob.core.windows.net" -> "{YOUR STORAGE ACCOUNT KEY}"))
```

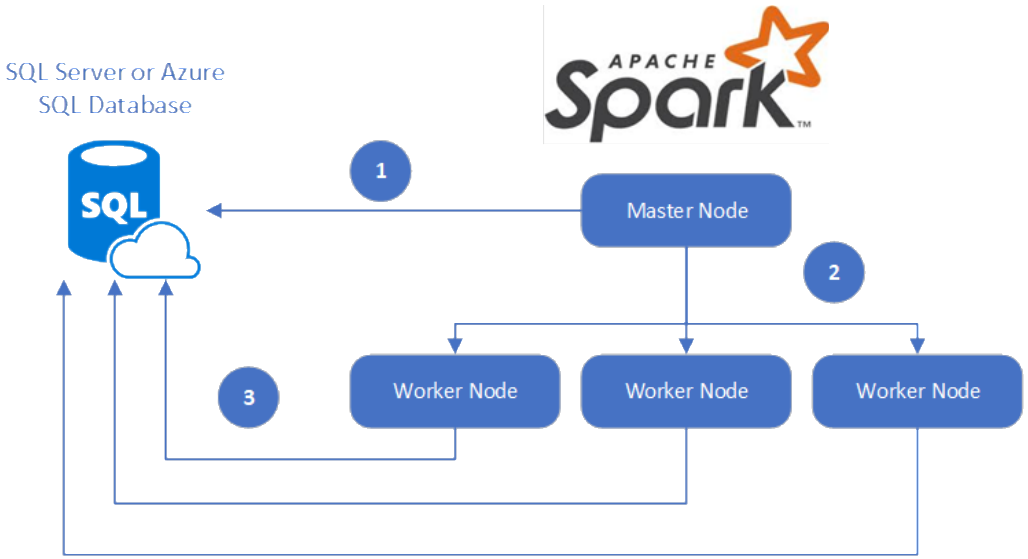
```
//Read or write via mount point
val df = spark.read.parquet("/mnt/wasb/OrdersParquet")
```



Integration experiences on Azure

How do I optimally load or query data into SQL Server?

Use [SQL Spark Connector] rather than JDBC

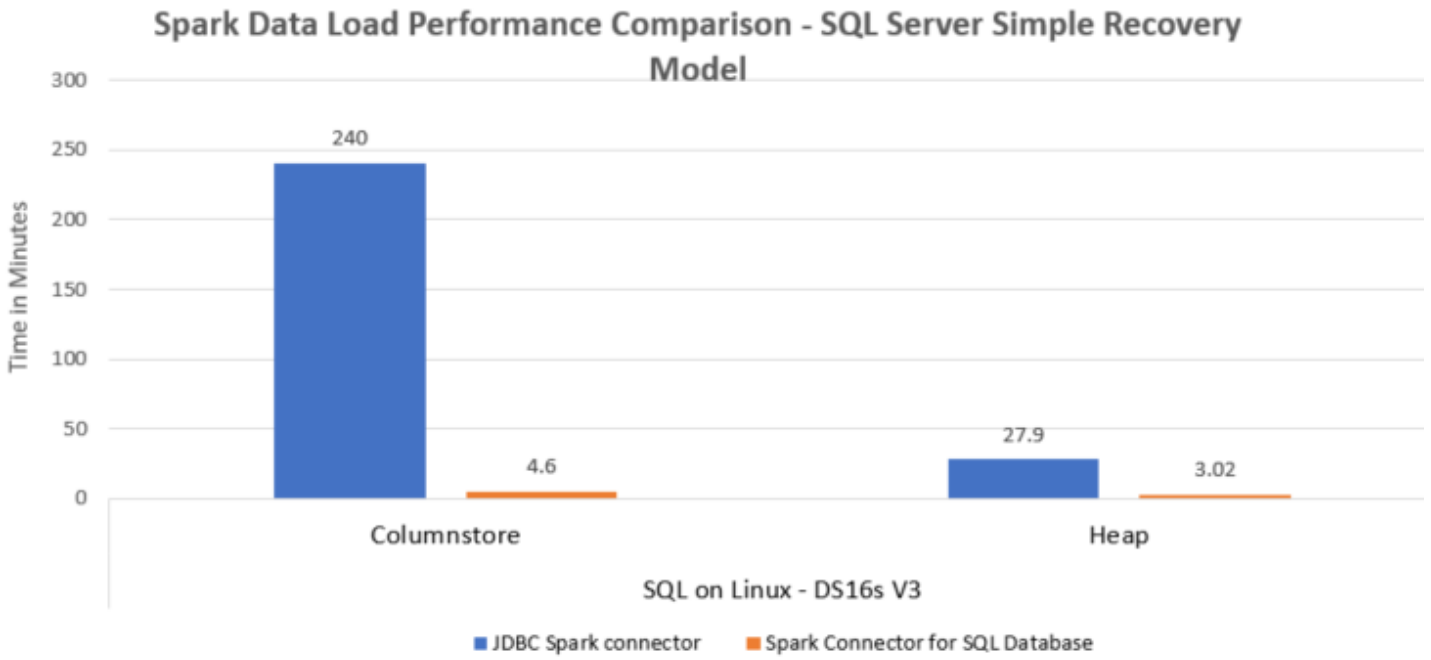


Load data to Azure SQL Database or SQL Server Instance

```
import com.microsoft.azure.sqldb.spark.config._
import com.microsoft.azure.sqldb.spark.connect._
import com.microsoft.azure.sqldb.spark.query._
import com.microsoft.azure.sqldb.spark._
import com.microsoft.azure.sqldb.spark.bulkcopy._

var bulkCopyMetadata = new BulkCopyMetadata
bulkCopyMetadata.addColumnMetadata(1, "o_orderkey", java.sql.Types.INTEGER, 0, 0)
bulkCopyMetadata.addColumnMetadata(2, "o_custkey", java.sql.Types.INTEGER, 0, 0)
//Only showing the first 2 columns being added to BulkCopyMetadata

val bulkCopyConfig = Config(Map(
  "url"               -> "server.westus.cloudapp.azure.com",
  "databaseName"      -> "testdb",
  "user"              -> "denzilr",
  "password"          -> "MyComplexPassword1!",
  "dbTable"           -> "dbo.orders",
  "bulkCopyBatchSize" -> "200000",
  "bulkCopyTableLock" -> "true",
  "bulkCopyTimeout"   -> "600"
))
dfOrders.bulkCopyToSqlDB(bulkCopyConfig)
```

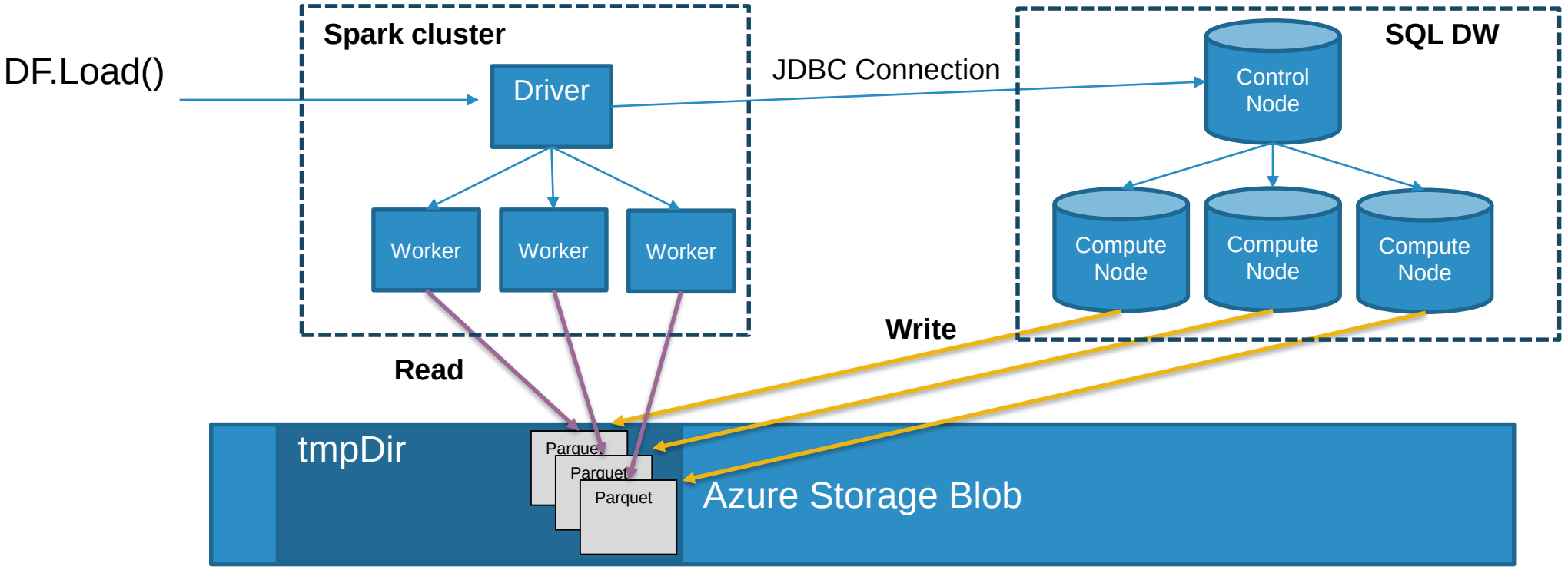
Sample bulk load performance

JDBC V/s SQL Connector comparison in data loading



- Specify TABLOCK for heaps
- Specify Batch size of 102,400 or greater for columnstore tables
- No TABLOCK for columnstore tables
- More details : [Turbo boost data loads from Spark using SQL Spark connector]

How do I read large datasets from SQL Data warehouse?



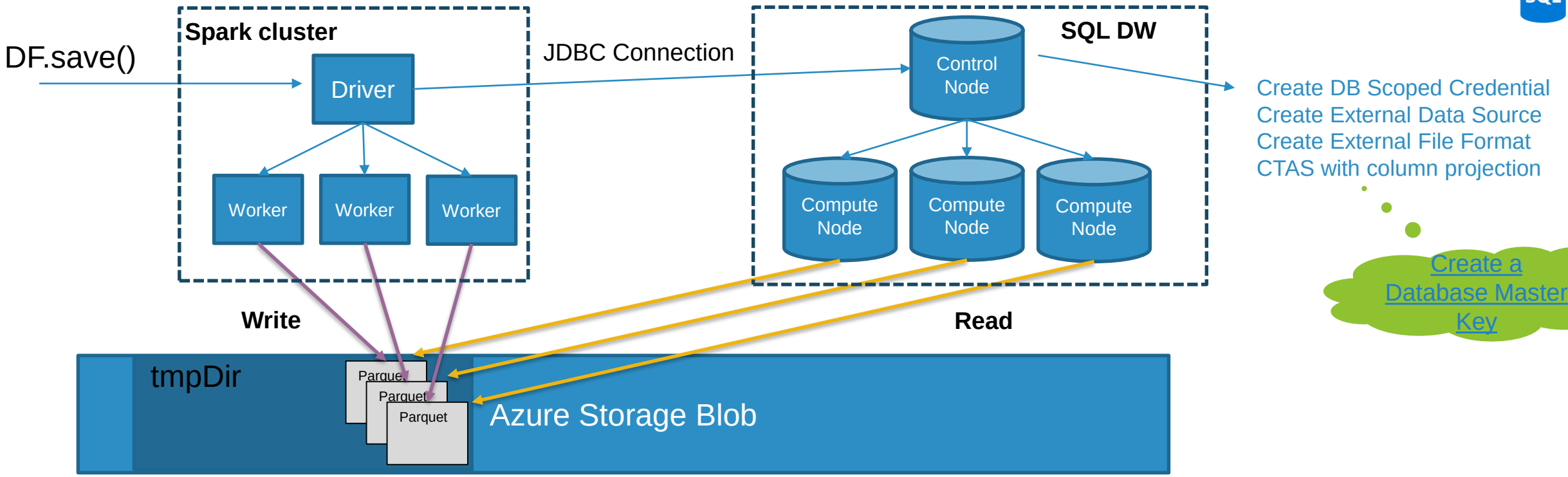

```
df = spark.read \
    .format("com.databricks.spark.sqldw")\
    .option("url", sqlDwUrlSmall) \
    .option("dbtable", "Dim_aircraft") \
    .option( "forward_spark_azure_storage_credentials","True") \
    .option("tempdir", tempDir) \
    .load()
```

JDBC connection string

DW Table

Azure Blob Storage directory

How do I load large datasets into SQL Data warehouse?

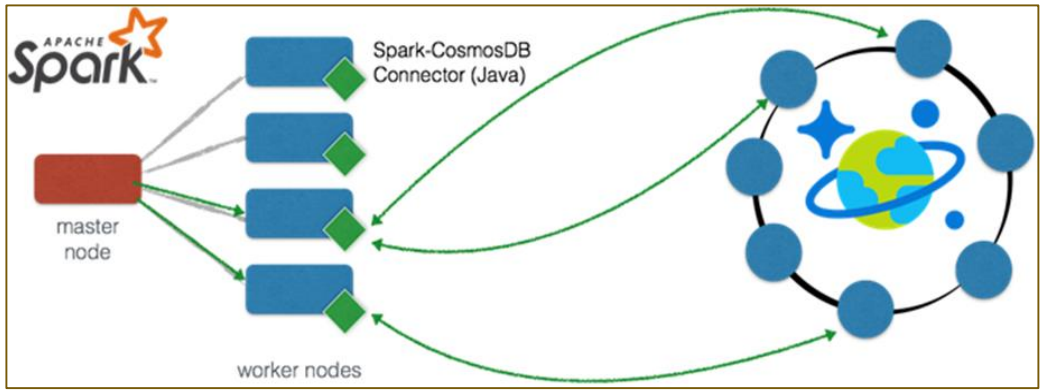



```
DF.write \
  .format("com.databricks.spark.sqldw")\
  .option("url", sqlDwUrlSmall)\
  .option("forward_spark_azure_storage_credentials", "true")\
  .option("tempdir", tempDir)\
  .option("dbtable", "DatabricksTable")\
  .option("table_option", "Distribution = HASH(Code), CLUSTERED INDEX (ID ASC)")\
  .option("preactions", "Declare @Starttime datetime2 = Getdate() Insert into eventlog values ('CTAS load: Start', @Starttime)") \
  .option("postactions", "Declare @EndTime Datetime2 = Getdate() Insert into eventlog values ('CTAS load: End ', @EndTime)") \
  .mode("overwrite")\
  .save()
```

Pre and Post Execution Steps

How do I optimally read data from CosmosDB?

- **With Spark to Azure ComosDB connector, Spark can now interact with all Cosmos DB data models: *Documents, Tables, and Graphs***
  - Exploits the native Azure Cosmos DB managed indexes
  - Utilizes push-down predicate filtering on fast-changing globally-distributed data




```
# Read Configuration
readConfig = {
  "Endpoint" : "https://mycosmosdb.documents.azure.com:443/",
  "Masterkey" : "my-master-key",
  "Database" : "AirlineDB",
  "preferredRegions" : "Central US;East US2",
  "Collection" : "flights",
  "SamplingRatio" : "1.0",
  "schema_samplesize" : "1000",
  "query_pagesize" : "2147483647",
  "query_custom" : "SELECT c.date, c.distance, c.origin
                    FROM c WHERE c.origin = 'SEA'"
}

# Connect via azure-cosmosdb-spark to create Spark DataFrame
flights = spark.read.format("com.microsoft.azure.cosmosdb.spark")
          .options(**readConfig).load()
flights.count()
```

How do I optimally bulk load data into CosmosDB?

- Use Bulk API
- Default Batch size of 500 is low. Consider increasing to 100,000
- Configure indexing to lazy (*note reads maybe inconsistent for a while)
- Monitor and Increase RU's as needed
- Try to map Spark partition setup to CosmosDB partitioning
- Use a larger page size to reduce network roundtrips

```
val writeConfig = Config(Map(
      "Endpoint" -> "https://myiotcosmos.documents.azure.com:443/",
      "Masterkey" -> "<your-key>",
      "Database" -> "airlinedata",
      "PreferredRegions" -> "West US;East US;",
      "Collection" -> "flightdata",
      "BulkImport" -> "true",
      "WritingBatchSize" -> "100000"
))

val writtenDf = rdf.write.mode("overwrite").cosmosDB(writeConfig)
```

Indexing Policy

```
{
    "indexingMode": "lazy",
    "automatic": true,
    "includedPaths": [
        {
            "path": "/*",
```

How do I build a streaming solution on Azure?

Connect to Azure Event Hub from a Spark Cluster using the Event Hub Spark Connector

1. Setup connection to Event Hub

```
val ehConnection = ConnectionStringBuilder("<your eventhub access key>")
                   .setEventHubName("MyIOTEvents")
                   .build

val eventHubConf = EventHubsConf(ehConnection)

val df = spark.readStream
         .format("eventhubs")
         .options(eventHubConf.toMap)
         .load()
```

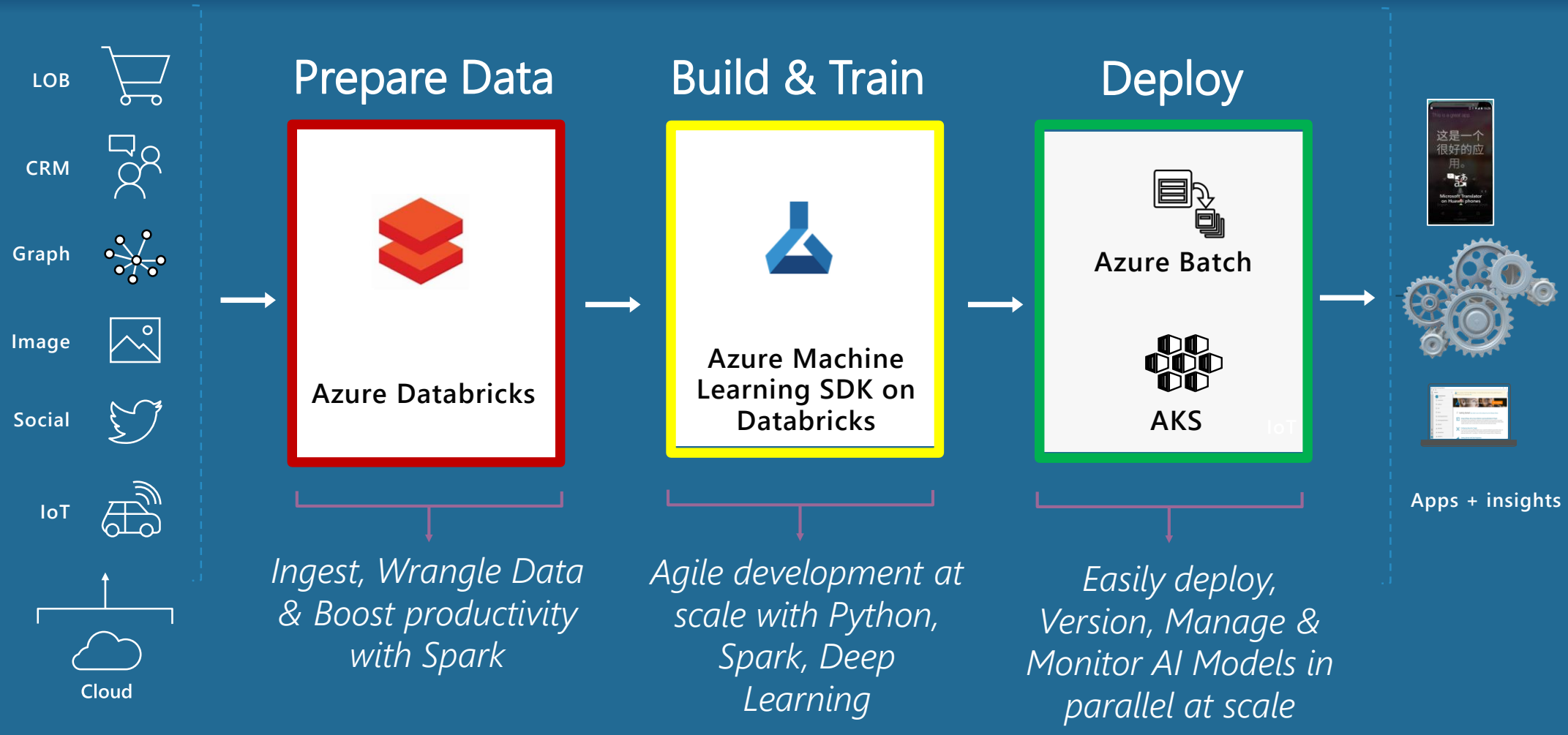
2. Read data from Event Hub

```
val eventhubdata = df.select($"body" cast "string")

val query1 = eventhubdata.writeStream
     .outputMode("append")
     .queryName("ehtable")
     .format("console")
     .start().awaitTermination()

val testData = spark.sql("select * from ehtable")
```

Event hubs for Kafka is in preview with Kafka API (1.0.0) calls

Event Hub auto-inflate feature can scale up to handle load increases

How do I operationalize my ML model?



How do I operationalize my ML model?

Azure ML API to serialize and deploy the model from your Databricks notebook

1. Provision Azure Kubernetes Service

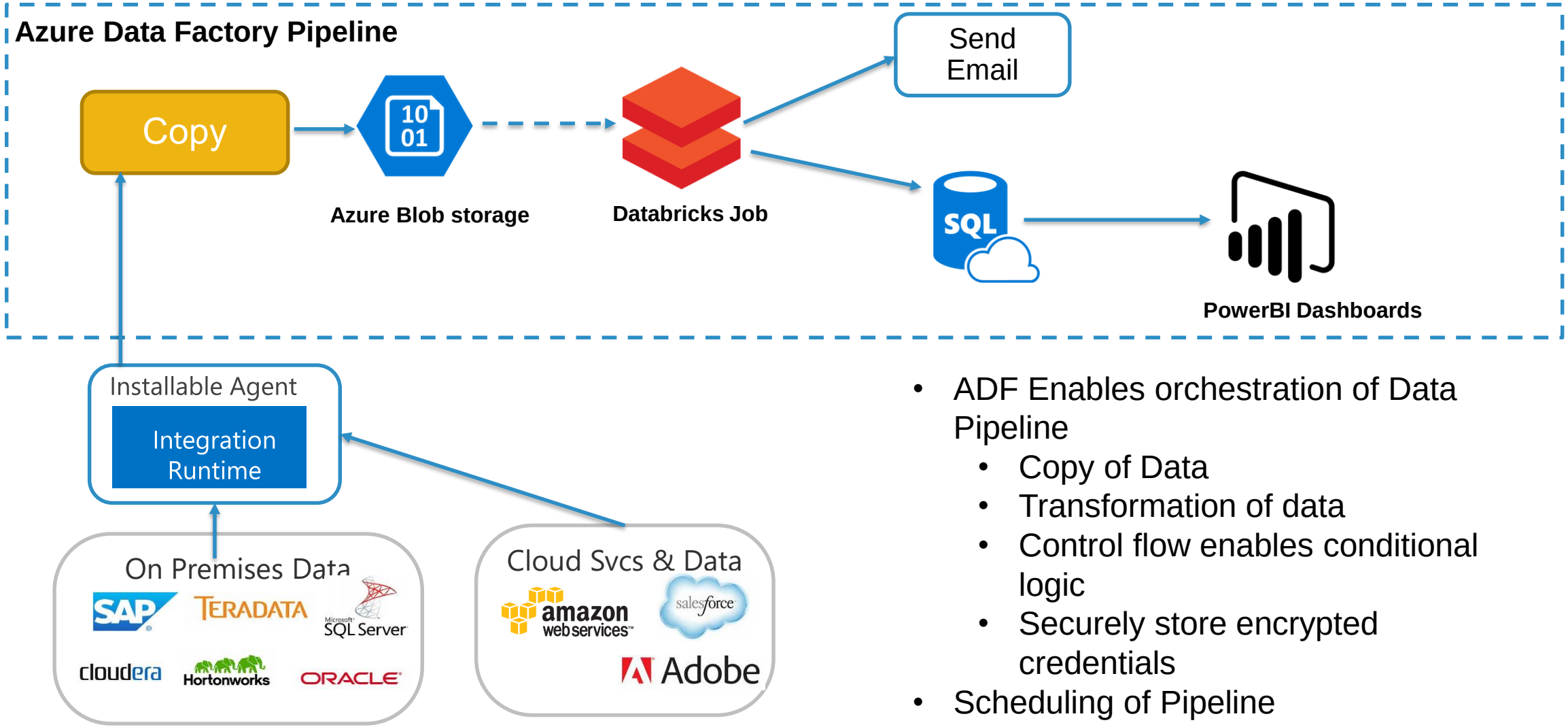
```
from azureml.core.webservice import AksWebservice

aks_config = AksWebservice.deploy_configuration(
    cpu_cores = 4,
    memory_gb = 8,
    cluster_min_nodes = 2, cluster_max_nodes = 4,
    description = 'Databricks Spark and AI Summit is awesome!!!')
```

Azure container services

2. Deploy the model web service

```
service_name = "sparkmlmodelserviceaks"
models = [model_name]
runtime = "spark-py"
conda_file = 'myenv_sparkml.yml'
driver_file = "score_sparkml.py"

webservice = AksWebservice(w, name=service_name)
webservice = w.deploy_webservice(
    name=service_name
    , models=models
    , runtime=runtime
    , conda_file=conda_file
    , deploy_config=aks_config
    , target=aks_compute
    #, image=image
    , driver=driver_file
    #, schema=schema_file
    )
webservice.wait_for_deployment(show_output=True)
```

How do I orchestrate my data pipeline?



- ADF Enables orchestration of Data Pipeline
  - Copy of Data
  - Transformation of data
  - Control flow enables conditional logic
  - Securely store encrypted credentials
- Scheduling of Pipeline

Best Practices

Security best practices

Databricks Workspace

- Provides most isolation
- Ensure User Access Control and Cluster Access control is enabled

Storage security

- Use different SAS tokens with different permission sets for Blob Storage
- Can mount different ADLS folders with different Access permissions
- Use ADLS SPNs for permissions on folders
- Encryption is default for all storage accounts
- Table ACLs exist only on clusters allowing python and SQL commands

Secrets Management

- Use Databricks secrets to store credentials needed in Spark conf file or connection strings
- Secure personal access tokens

Performance and limits

Performance Considerations

- VM Size limits (Cores, Networking, IO)
- VM SSD size and shuffle/DBIO Caching
- GPU optimized VM's for AI and Deep learning models
- Storage IOPs limits and monitoring for Azure storage
- Region Awareness (co-location with other Azure Services)

Service scalability

- Azure Event Hub auto-inflate settings
- Databricks cluster auto-scale
- CosmosDB request units for increased throughput
- Managing Data warehouse compute
- Azure SQL database service tier

Quotas and Limits

- Default subscription quotas on large clusters (Cores/IP)

Operational Best practices

Cluster Logs location

- Custom log delivery to a DBFS location
- Use a mount point to store logs on Azure blob storage

DR site

- Workspace API export, instantiate new workspace
- Storage replication

Monitoring

- Monitor DBIO cache size and pick the appropriate worker node types
- Can enable OMS monitoring with init script associating cluster VM's to an OMS workspace

Sample OMS monitoring chart

Q&A

Thank You!!!