Sponsors for Global Azure Bootcamp

Thanks !

Microsoft

The National Security Group

Insuring your world.

GDH

GOVERNMENT SERVICES

ALFA®

INSURANCE

doozer

SOFTWARE

Azure HDInsight

- Azure Bootcamp -

2017. 4. 22

Ingyu Lee
(inlee@troy.edu)

https://github.com/Azure-Bootcamp-Troy/AzureHDInsight

Contents

- Target Audience
  - ✓ Database / BI Professional.
  - ✓ Data Scientists / Analysts with some technical experience.

- Prerequisites
  - ✓ Familiarity with database concepts and basic SQL query syntax.
  - ✓ Basic understanding of Unix commands.
  - ✓ Familiarity with programming fundamentals.

- Outline

  P1: Getting Started with HDInsight.
  P2: Processing Big Data with Hive.
  P3: Going Beyond Hive with Pig and Python.
  P4: Using HBase for NoSQL Data

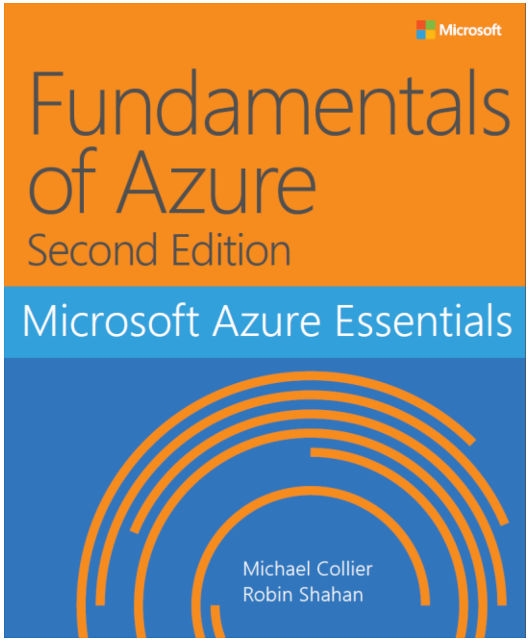
- Hands-on Labs
  - ✓ Free trial available.
  - ✓ Microsoft Azure Subscription.

- edX (Microsoft)
  - ✓ DAT202.1x Processing Big Data with Hadoop in Azure HDInsight
  - ✓ DAT202.2x Implementing Real-Time Analytics with Hadoop in Azure HDInsight
  - ✓ DAT202.3x Implementing Predictive Analytics with Spark in Azure HDInsight

- MVA (Microsoft Virtual Academy)
  - ✓ Big Data Analytics with HDInsight
  - ✓ Implementing Big Data Analysis Jump Start

- Coursera (Big Data Specialization)
  - ✓ Hadoop Platform and Application Framework
  - ✓ Introduction to Big Data Analytics

Free eBooks



Microsoft

Fundamentals of Azure

Second Edition

Microsoft Azure Essentials

Michael Collier
Robin Shahan

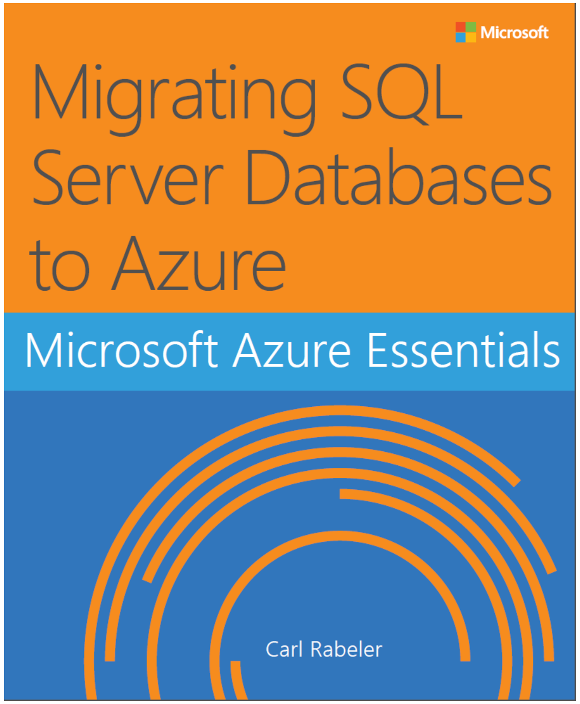
Microsoft

Introducing

Microsoft Azure HDInsight

Technical Overview

Avkash Chauhan, Valentine Fontama,
Michele Hart, Wee Hyong Tok, Buck Woody



Microsoft

Migrating SQL Server Databases to Azure

Microsoft Azure Essentials

Carl Rabeler

http://www.microsoftvirtualacademy.com/ebooks

I Introduction to Big Data and HDInsight

1. What is Big Data?

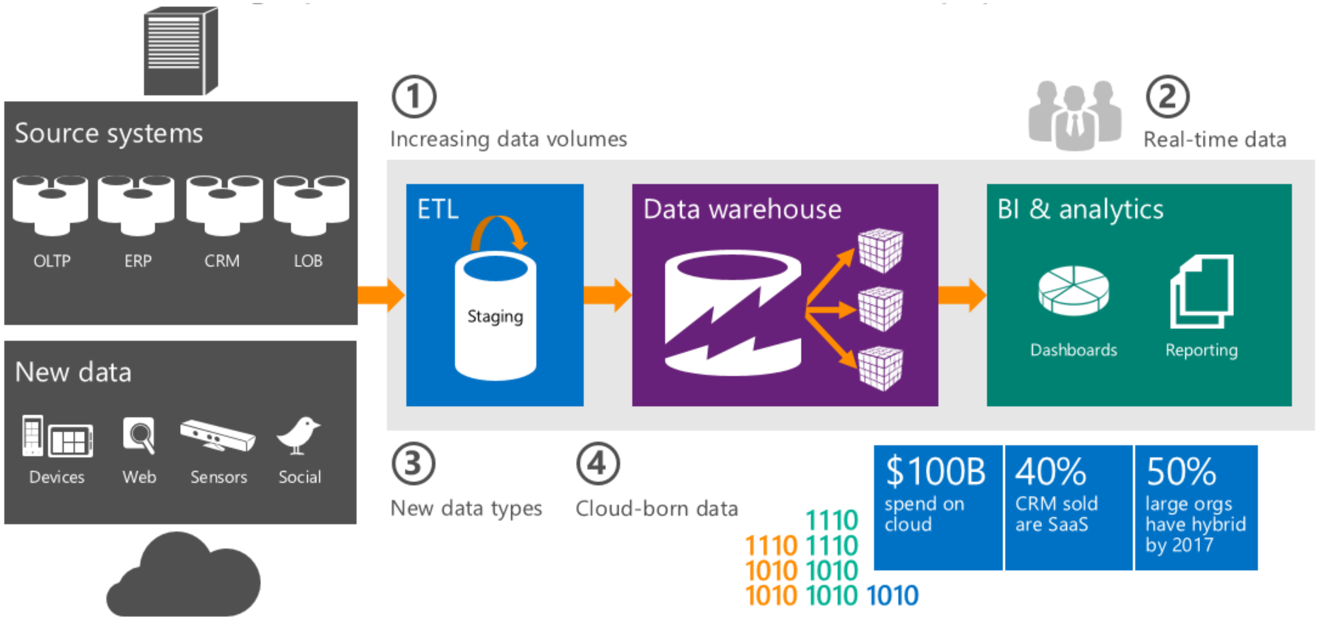
    a. Data that is too large or complex for analysis in traditional relational databases.

    b. Typified by the “3 Vs”

        a. Volume: Huge amounts of data to process.

        b. Variety: A mixture of structured and unstructured data.

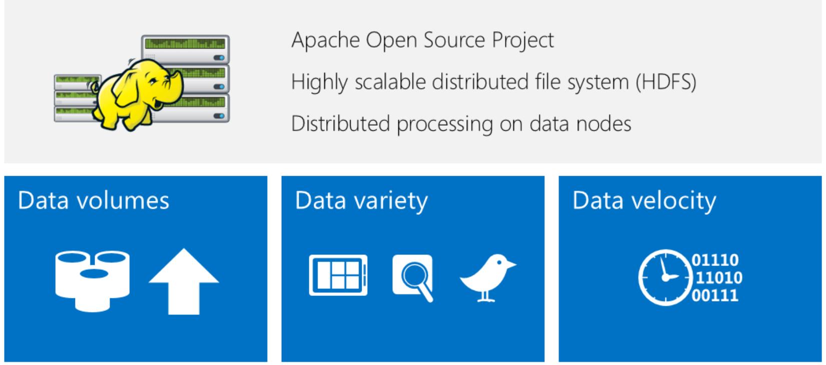
        c. Velocity: New data generated extremely frequently.



Breaking Point of Traditional Approach

I Introduction to Big Data and HDInsight: What is Hadoop?

1. What is Hadoop?
   a. Open source distributed data processing cluster.
   b. Data processed in Hadoop Distributed File System (HDFS).
   c. Resource Management is performed by YARN.



Apache Open Source Project

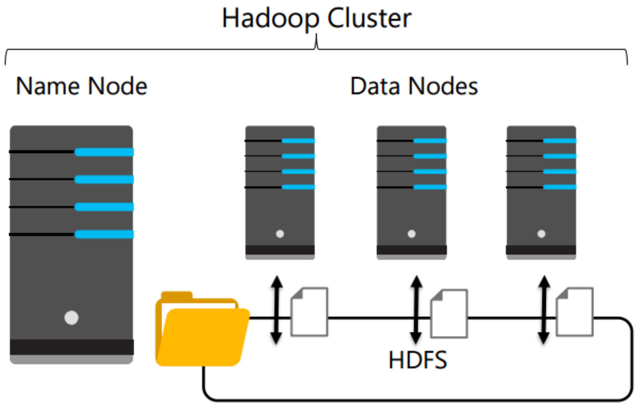
Highly scalable distributed file system (HDFS)

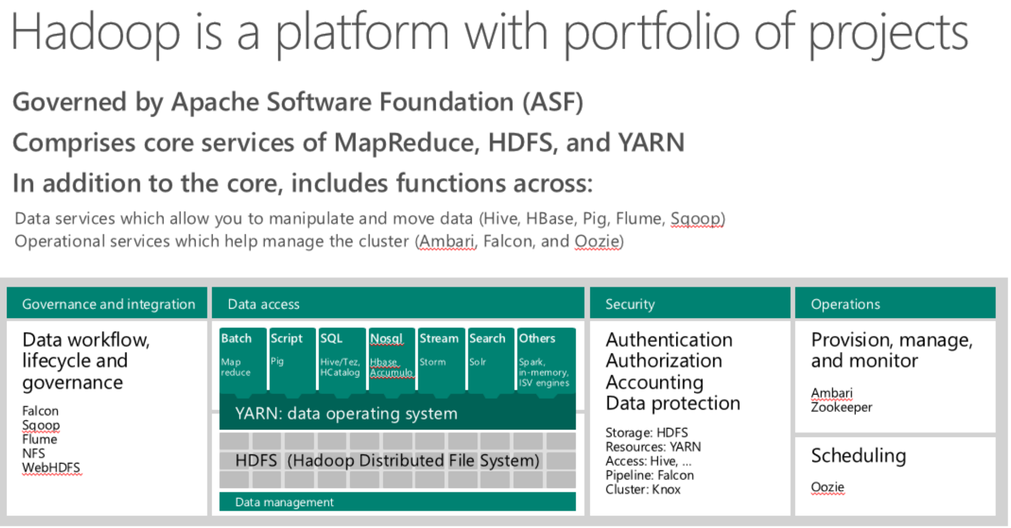
Distributed processing on data nodes

Data volumes

Data variety

Data velocity





Hadoop is a platform with portfolio of projects

Governed by Apache Software Foundation (ASF)

Comprises core services of MapReduce, HDFS, and YARN

In addition to the core, includes functions across:

Data services which allow you to manipulate and move data (Hive, HBase, Pig, Flume, Sqoop)
Operational services which help manage the cluster (Ambari, Falcon, and Oozie)

Governance and integration

Data workflow, lifecycle and governance

Falcon
Sqoop
Flume
NFS
WebHDFS

Data access

Batch: Map reduce | Script: Pig | SQL: Hive/Tez, HCatalog | Nosql: Hbase, Accumulo | Stream: Storm | Search: Solr | Others: Spark, in-memory, ISV engines

YARN: data operating system

HDFS (Hadoop Distributed File System)

Data management

Security

Authentication
Authorization
Accounting
Data protection

Storage: HDFS
Resources: YARN
Access: Hive, ...
Pipeline: Falcon
Cluster: Knox

Operations

Provision, manage, and monitor

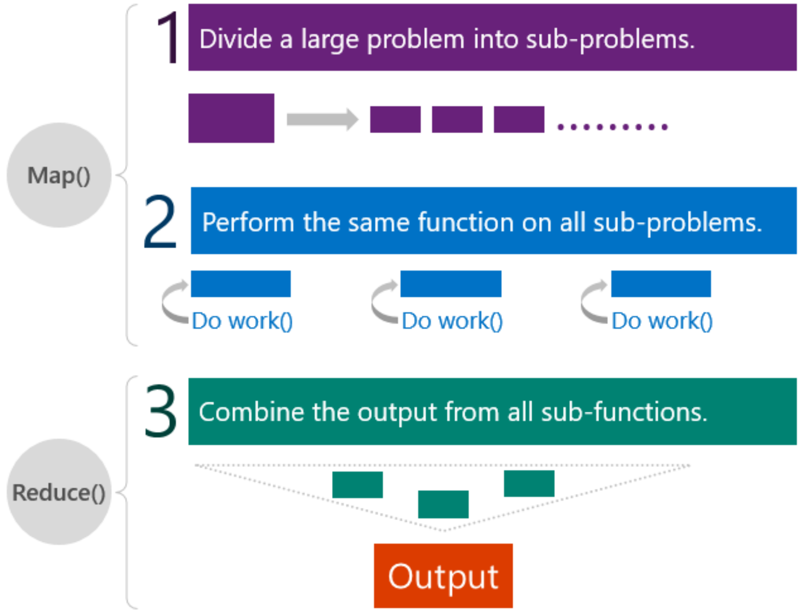
Ambari
Zookeeper

Scheduling

Oozie

I Hadoop MapReduce

- Programming framework (library and runtime) for analysing datasets stored in HDFS
- Composed of user-supplied Map and Reduce functions:
  - Map() - subdivide and conquer
  - Reduce() - combine and reduce cardinality



Map()

1 Divide a large problem into sub-problems.

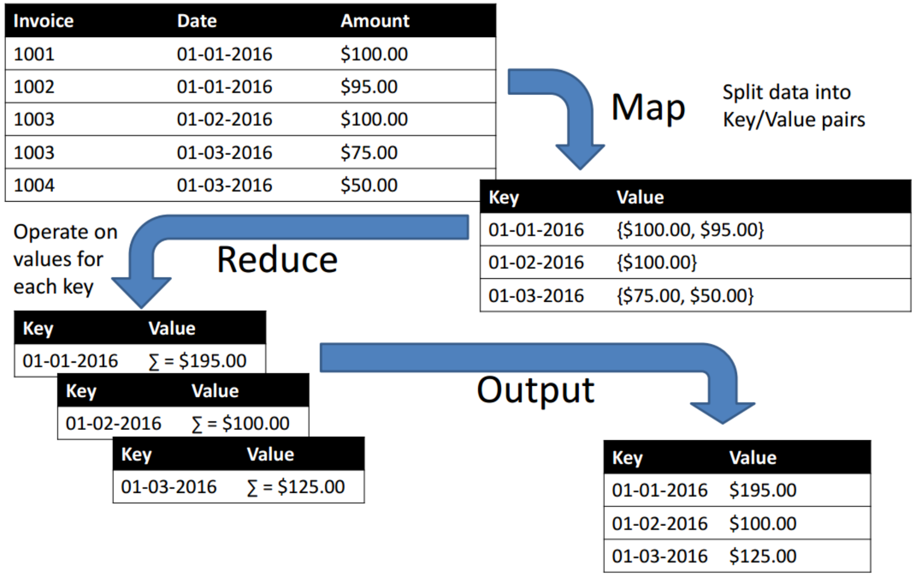
2 Perform the same function on all sub-problems.

Do work() Do work() Do work()

Reduce()

3 Combine the output from all sub-functions.

Output



Invoice	Date	Amount
1001	01-01-2016	$100.00
1002	01-01-2016	$95.00
1003	01-02-2016	$100.00
1003	01-03-2016	$75.00
1004	01-03-2016	$50.00

Map

Split data into Key/Value pairs

Key	Value
01-01-2016	{$100.00, $95.00}
01-02-2016	{$100.00}
01-03-2016	{$75.00, $50.00}

Reduce

Operate on values for each key

Key	Value
01-01-2016	∑ = $195.00

Key	Value
01-02-2016	∑ = $100.00

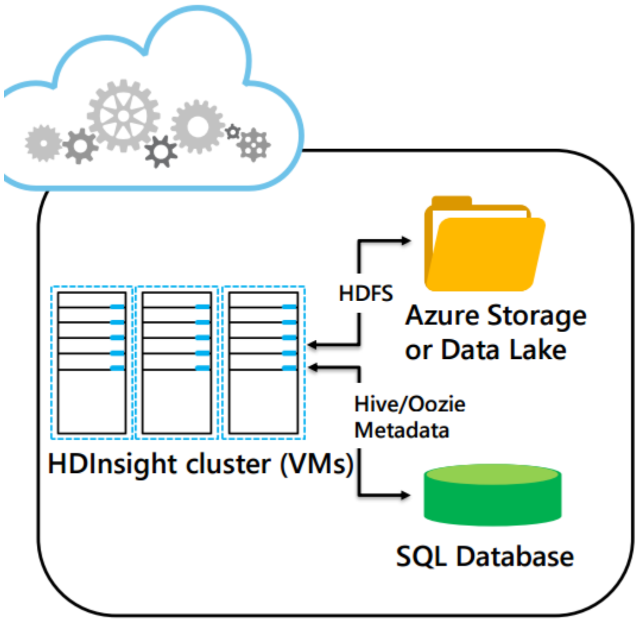
Key	Value
01-03-2016	∑ = $125.00

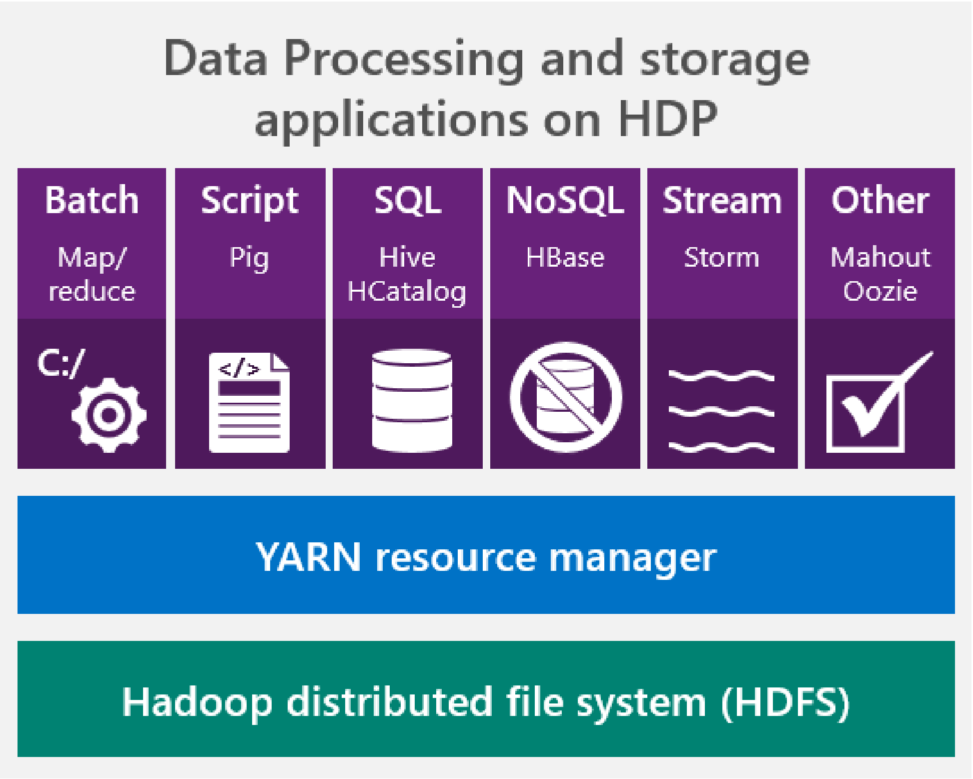
Output

Key	Value
01-01-2016	$195.00
01-02-2016	$100.00
01-03-2016	$125.00

I Introduction to Big Data and HDInsight

1. What is HDInsight?
   a. Apache Hadoop on Azure
      ✓ Hortonworks HDP on Azure VMs.
   b. Azure Storage or Azure Data Lake provides the HDFS layer.
   c. Azure SQL Database stores metadata.





Data Processing and storage applications on HDP

Batch	Script	SQL	NoSQL	Stream	Other
Map/ reduce	Pig	Hive HCatalog	HBase	Storm	Mahout Oozie

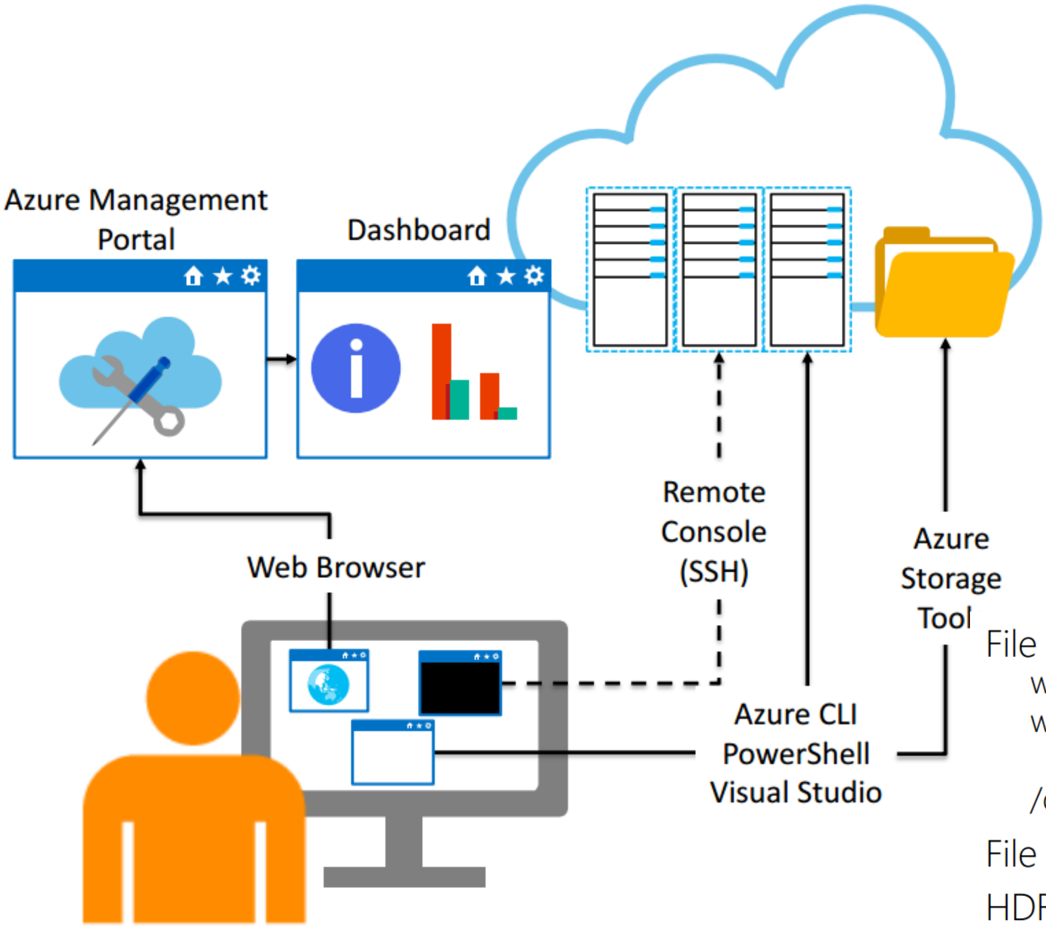
YARN resource manager

Hadoop distributed file system (HDFS)

- Choose from a range of Hadoop-compatible tools
- Run resource-intensive apps such as HBase or Storm on a separate cluster
- Ensure better ROI by checking query results

Demo: Provisioning an HDInsight Cluster

I HDInsight: What client tools can I use?



File paths can be referenced using WASB(S) or native syntax

wasb://*container*@*account*.blob.core.windows.net/data/logs/file.txt

wasb:///data/logs/file.txt (default storage account and container)

or

/data/logs/file.txt

File paths are <u>case-sensitive</u>

HDFS shell commands

ls (list)

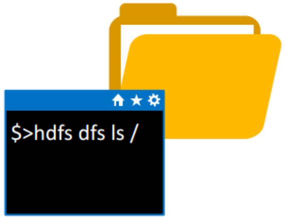
cp and **mv** (copy and move)

mkdir (make directory)

rm and **rm –r** (remove and remove recursive)

put and **get** (transfer files between local file system and HDFS)

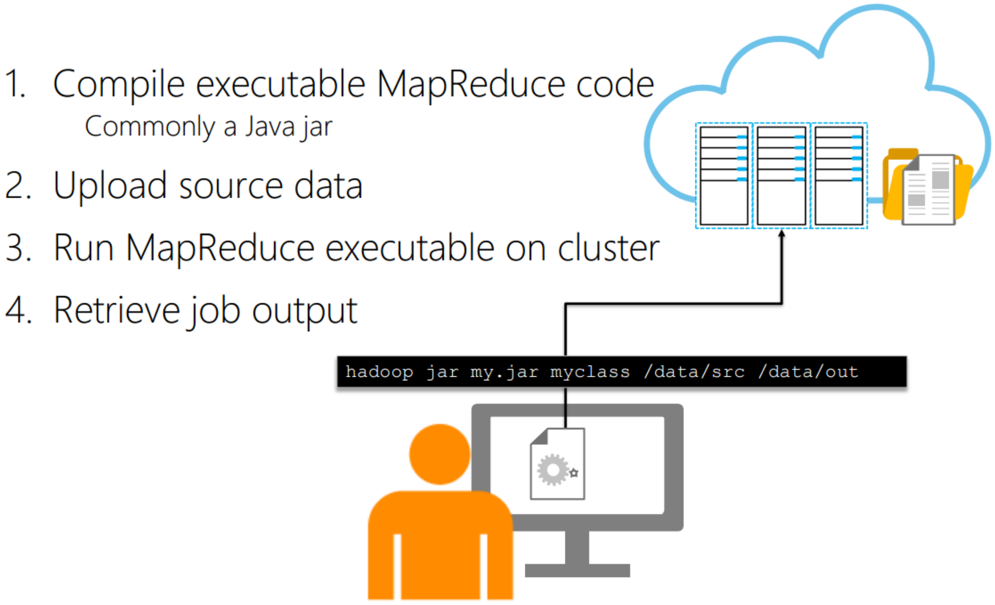
text, **cat**, and **tail** (display contents of file)



```
$>hdfs dfs ls /
```

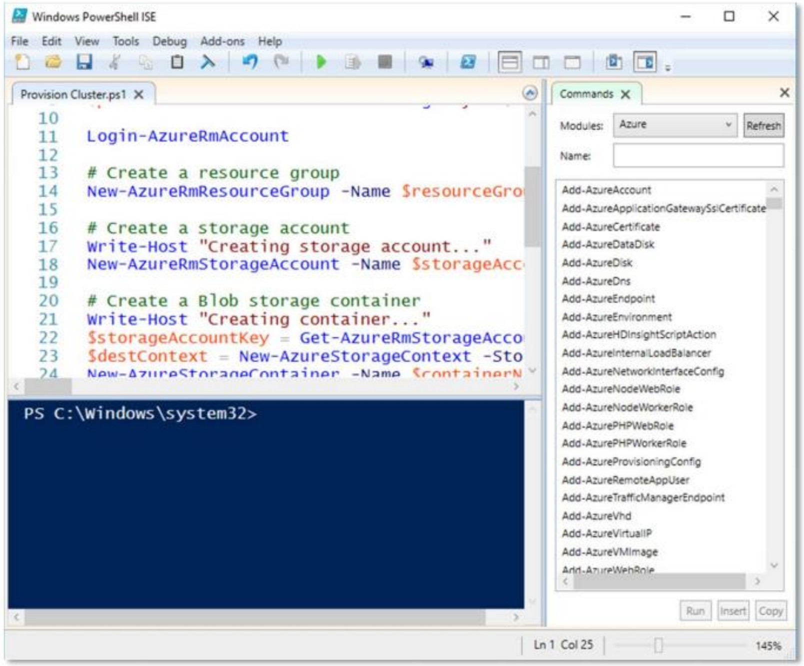
Demo: Using HDInsight Client Tools in Windows

I HDInsight: How do I Run a MapReduce Job



1. Compile executable MapReduce code
   Commonly a Java jar
2. Upload source data
3. Run MapReduce executable on cluster
4. Retrieve job output

```
hadoop jar my.jar myclass /data/src /data/out
```



```
Login-AzureRmAccount

# Create a resource group
New-AzureRmResourceGroup -Name $resourceGro

# Create a storage account
Write-Host "Creating storage account..."
New-AzureRmStorageAccount -Name $storageAcc

# Create a Blob storage container
Write-Host "Creating container..."
$storageAccountKey = Get-AzureRmStorageAcco
$destContext = New-AzureStorageContext -Sto
New-AzureStorageContainer -Name $containerN
```

```
PS C:\Windows\system32>
```

Add-AzureAccount
Add-AzureApplicationGatewaySslCertificate
Add-AzureCertificate
Add-AzureDataDisk
Add-AzureDisk
Add-AzureDns
Add-AzureEndpoint
Add-AzureEnvironment
Add-AzureHDInsightScriptAction
Add-AzureInternalLoadBalancer
Add-AzureNetworkInterfaceConfig
Add-AzureNodeWebRole
Add-AzureNodeWorkerRole
Add-AzurePHPWebRole
Add-AzurePHPWorkerRole
Add-AzureProvisioningConfig
Add-AzureRemoteAppUser
Add-AzureTrafficManagerEndpoint
Add-AzureVhd
Add-AzureVirtualIP
Add-AzureVMImage
Add-AzureWebRole

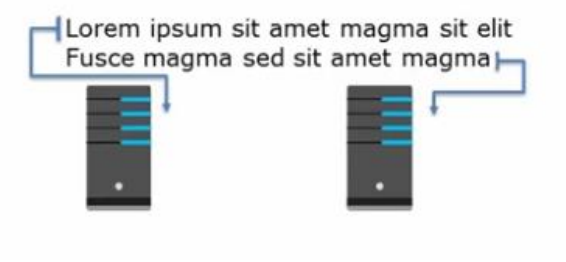
Demo: Run a MapReduce Job from Windows

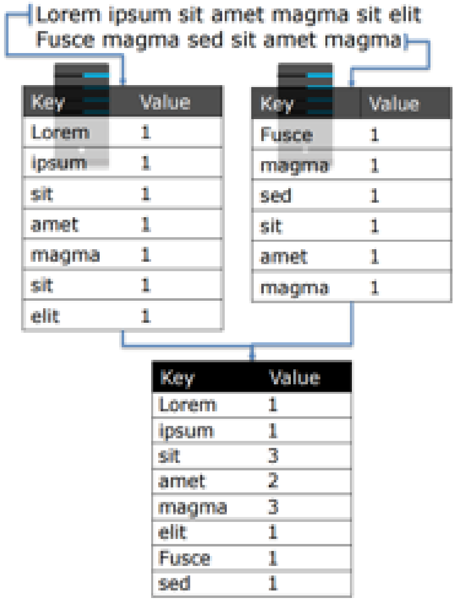
1. The Azure PowerShell module includes cmdlets to work with Azure services, including HDInsight
2. Use PowerShell to
   a. Provision HDInsight Clusters
   b. Upload/download files
   c. Submit jobs
   d. Manage cluster resources

Demo: Using PowerShell with HDInsight

I Introduction to Big Data and HDInsight: Example

1. Word Count (The “Hello World” of MapReduce)
   a. Source text is divided among data nodes.
   b. Map phase generates key/value pairs with words as keys and placeholder values of 1.
   c. Reduce phase aggregates values for each key by adding the values for each word.


Lorem ipsum sit amet magma sit elit
Fusce magma sed sit amet magma


Lorem ipsum sit amet magma sit elit
Fusce magma sed sit amet magma

Key	Value
Lorem	1
ipsum	1
sit	1
amet	1
magma	1
sit	1
elit	1

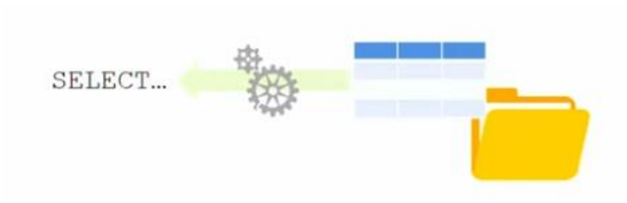
Key	Value
Fusce	1
magma	1
sed	1
sit	1
amet	1
magma	1

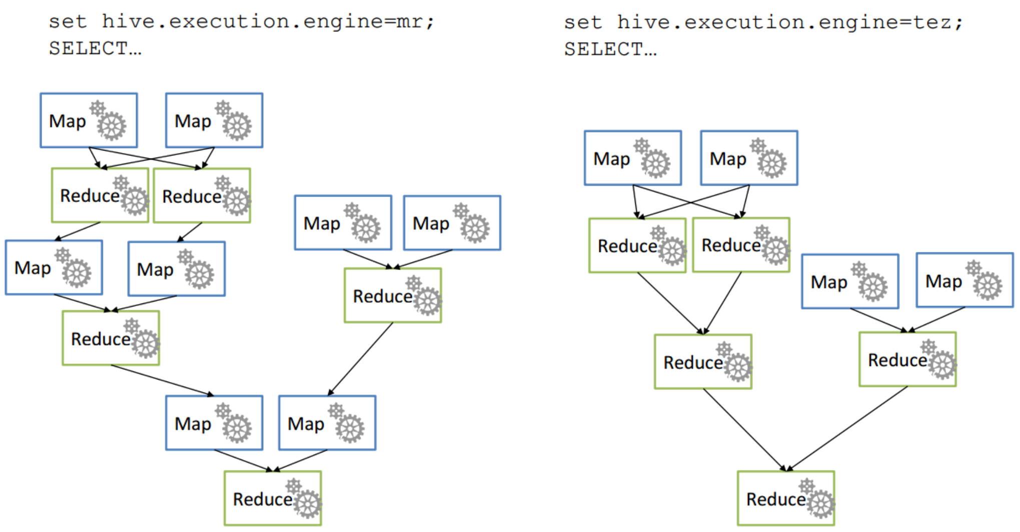
Key	Value
Lorem	1
ipsum	1
sit	3
amet	2
magma	3
elit	1
Fusce	1
sed	1

```
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
       private final static IntWritable one = new IntWritable(1);
       private Text word = new Text();
       public void map(LongWritable key, Text value, Context context) {
           String line = value.toString();
           StringTokenizer tokenizer = new StringTokenizer(line);
           while (tokenizer.hasMoreTokens()) {
               word.set(tokenizer.nextToken());
               context.write(word, one);
           }
       }
}

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
     public void reduce(Text key, Iterable<IntWritable> values, Context context){
       int sum = 0;
       for (IntWritable val : values) {
               sum += val.get();
       }
       context.write(key, new IntWritable(sum));
     }
}
```

II Processing Big Data with Hive: What is Hive?

1. Apache HIVE is a data warehouse system for Hadoop.
2. A metadata service that projects tabular schemas over folders.
3. Enables the contents of folders to be queried as tables, using SQL-like query semantics, HiveQL.
4. Queries are translated into jobs
   a. Execution engine can be Tez or MapReduce

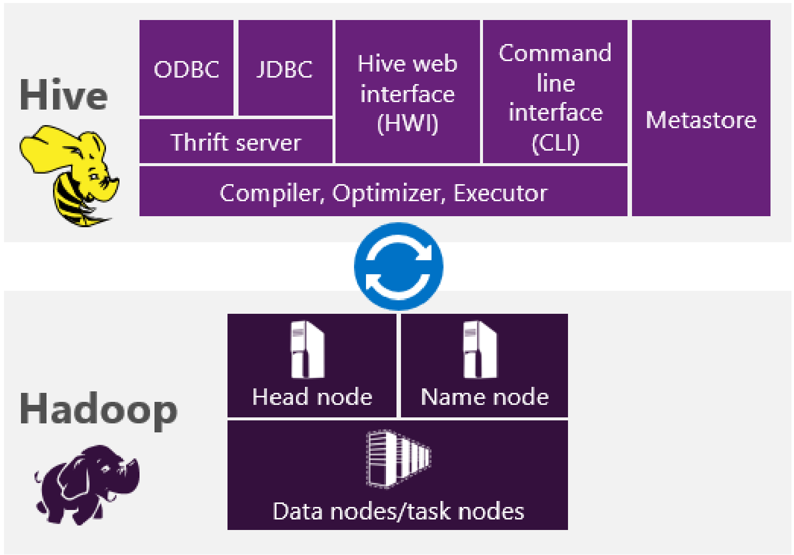




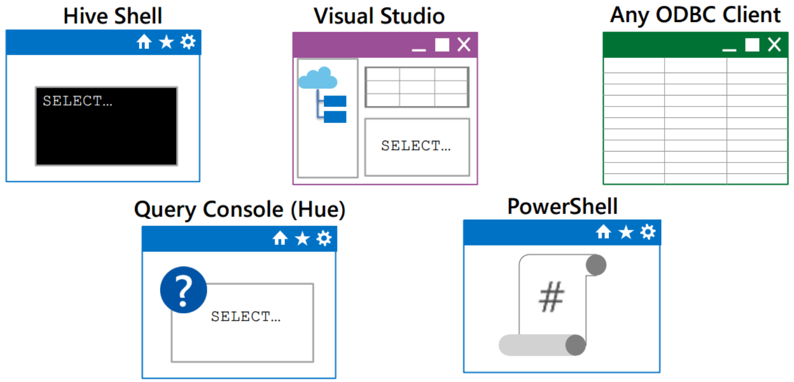
```
set hive.execution.engine=mr;
SELECT…
```

```
set hive.execution.engine=tez;
SELECT…
```

MapReduce vs. Tez

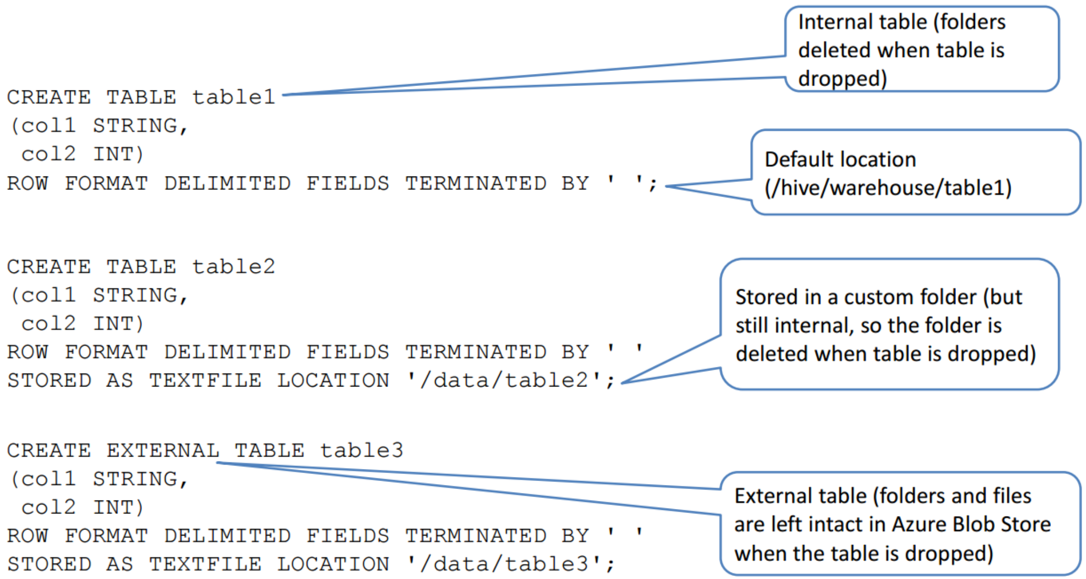


Hive client tools include…



II Hive: How do I create and load Hive tables?

1. Use the CREATE TABLE HiveQL statement
   a. Defines schema metadata to be projected onto data in a folder when the table is queried (not when it is created)
2. Specify file format and file location
   a. Defaults to textfile format in the <database>/<table-name> folder
      - ✓ Default database is in /hive/warehouse
      - ✓ Create additional databases using CREATE DATABASE
3. Create internal or external tables
   a. Internal tables manage the lifetime of the underlying folders
   b. External tables are managed independently from folders



```
CREATE TABLE table1
(col1 STRING,
 col2 INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';

CREATE TABLE table2
(col1 STRING,
 col2 INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION '/data/table2';

CREATE EXTERNAL TABLE table3
(col1 STRING,
 col2 INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION '/data/table3';
```

Internal table (folders deleted when table is dropped)

Default location (/hive/warehouse/table1)

Stored in a custom folder (but still internal, so the folder is deleted when table is dropped)

External table (folders and files are left intact in Azure Blob Store when the table is dropped)

II Hive: Hive data types

1. Numeric
   a. Integers: TINYINT, SMALLINT, INT, BIGINT
   b. Fractional: FLOAT, DOUBLE, DECIMAL
2. Character
   a. STRING, VARCHAR, CHAR
3. Date/Time
   a. TIMESTAMP
   b. DATE
4. Special
   a. BOOLEAN, BINARY, ARRAY, MAP, STRUCT, UNIOTYPE

1. Save data files in table folders (or create table on existing files!)

PUT myfile.txt /data/table1

2. Use the LOAD statement

LOAD DATA (LOCAL) INPATH ‘/data/source’ INTO TABLE MyTable;

3. User the INSERT statement

INSERT INTO TABLE Table2

SELECT Col1, UPPER(Col2)

FROM Table1;

4. Use a CREATE TABLE AS SELECT (CTAS) statement

CREATE TABLE Table3

ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘\t’

STORED AS TEXTFILE LOCATION ‘/data/summarytable’

AS

SELECT Col1, SUM(Col2) As Total

FROM Table1

GROUP BY Col1;

II Hive: How do I query Hive Tables?

1. Query data using the SELECT HiveQL statement

```
SELECT Col1, SUM(Col2) AS TotalCol2
FROM MyTable
WHERE Col3 = ‘ABC’ AND Col4 < 10
GROUP BY Col1
ORDER BY Col4;
```

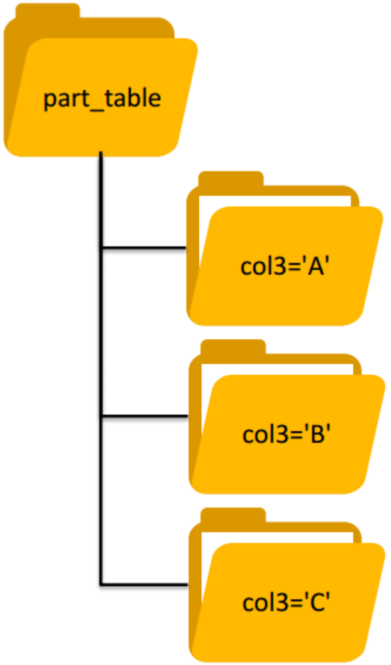
2. Hive translates the query into jobs and applies the table schema to the underlying data files

3. Views are named queries that abstract underlying tables

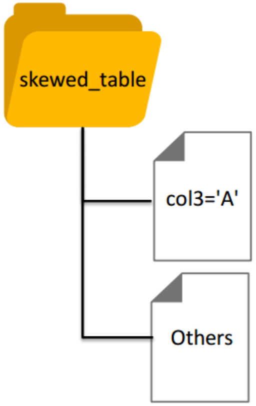
```
CREATE VIEW vSummarizeData
AS
SELECT col1, SUM(col2) AS TotalCol2
FROM mytable
GROUP BY col1;

SELECT col1, TotalCol2
FROM vSummarizeData
```

II Hive: Partitioning, Skewing, and Clustering Tables

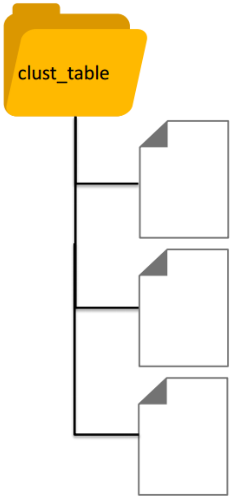
```
CREATE TABLE part_table
(col1 INT,
 col2 STRING)
PARTITIONED BY (col3 STRING);

INSERT INTO TABLE part_table PARTITION(col3='A')
SELECT col1, col2
FROM stg_table
WHERE col3 = 'A';

SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode=nonstrict;

INSERT INTO TABLE part_table PARTITION(col3)
SELECT col1, col2, col3
FROM stg_table;
```



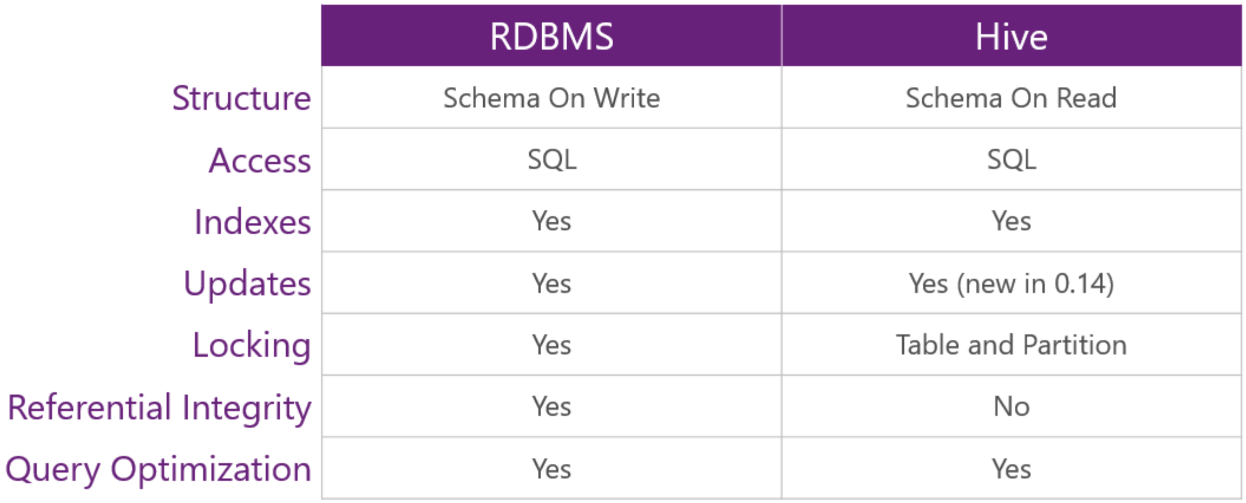
```
CREATE TABLE skewed_table
(col1 INT,
 col2 STRING,
 col3 STRING)
SKEWED BY (col3) ON ('A') [STORED AS DIRECTORIES];


INSERT INTO TABLE skewed_table
SELECT col1, col2, col3
FROM stg_table;
```



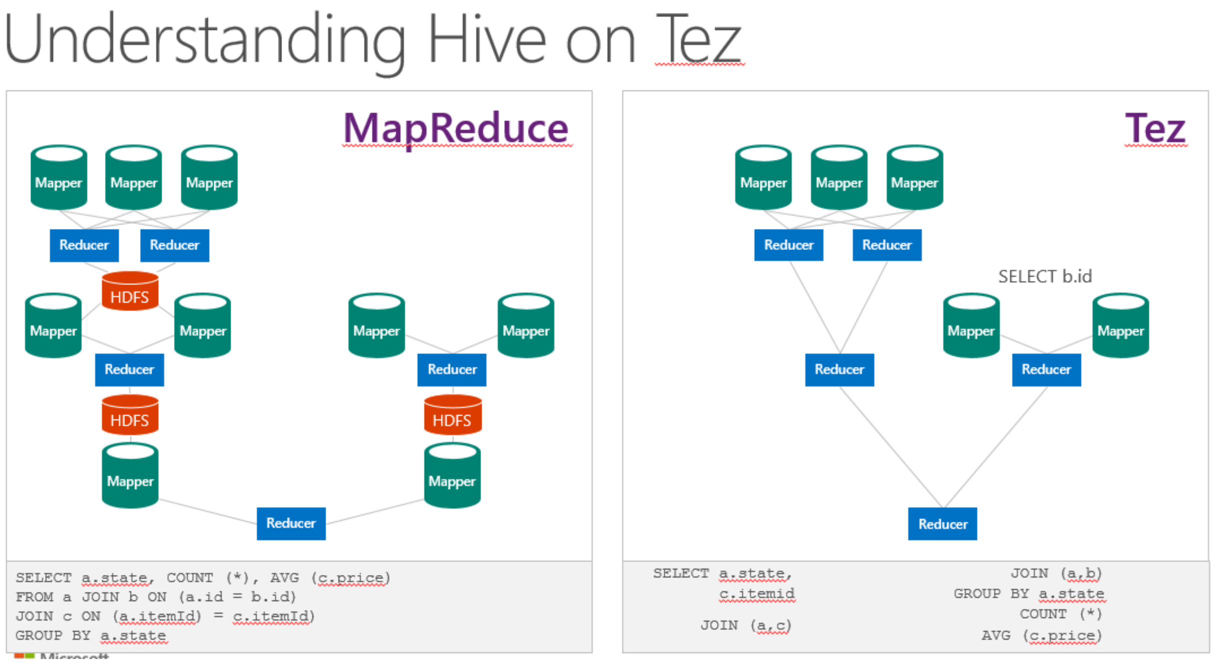
```
CREATE TABLE clust_table
(col1 INT,
 col2 STRING,
 col3 STRING)
CLUSTERED BY (col3) INTO 3 BUCKETS;


INSERT INTO TABLE clust_table
SELECT col1, col2, col3
FROM stg_table;
```



II Hive Comparing RDBMS and Hive



	RDBMS	Hive
Structure	Schema On Write	Schema On Read
Access	SQL	SQL
Indexes	Yes	Yes
Updates	Yes	Yes (new in 0.14)
Locking	Yes	Table and Partition
Referential Integrity	Yes	No
Query Optimization	Yes	Yes



Understanding Hive on Tez

MapReduce

Tez

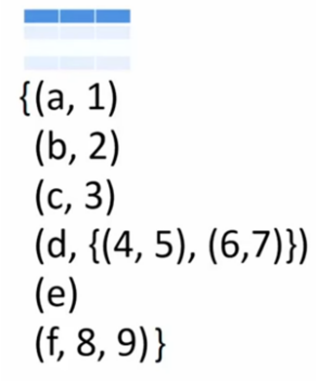
```
SELECT a.state, COUNT (*), AVG (c.price)
FROM a JOIN b ON (a.id = b.id)
JOIN c ON (a.itemId) = c.itemId)
GROUP BY a.state
```

```
SELECT a.state,                JOIN (a,b)
       c.itemid          GROUP BY a.state
                               COUNT (*)
       JOIN (a,c)          AVG (c.price)
```

Microsoft

III Beyond Hive: What is Pig?

1. Pig is a high level scripting language used with Hadoop.
2. Pig performs a series of transformations to data relations based on Pig Latin statements.
3. Relations are loaded using schema on read semantics to project table structure at runtime.
4. You can run Pig Latin statements interactively in the Grunt shell or save a script file and run them as a batch.
5. A relation is an outer bag
    a. A bag is a collection of tuples
    b. A tuple is an ordered set of fields
    c. A field is a data item
6. A field can contain an inner bag
7. A bag can contain tuples with nonmatching schema



```
{(a, 1)
 (b, 2)
 (c, 3)
 (d, {(4, 5), (6,7)})
 (e)
 (f, 8, 9)}
```

Pig—Where it fits

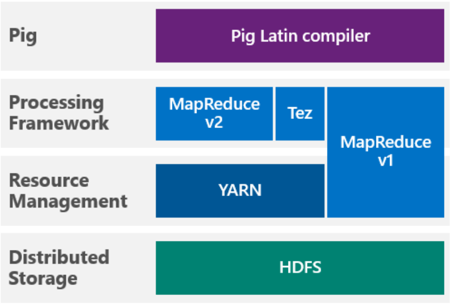
Pig is designed to perform long series of data operations like:

- Extract-transform-load (ETL) data pipelines
- Research on raw data
- Iterative data processing

Key benefits

- **Extensible.** Create custom functions to meet your particular processing requirements
- **Easy to program.** Simplify and encode complex tasks involving interrelated data transformations as data flow sequences—easily write and maintain huge tasks
- **Self-optimizing.** The system automatically optimizes execution of Pig jobs so you can focus on semantics

What makes Pig so special?



Pigs are special because:

- **Pigs eat anything.** Pig can operate on data whether it has metadata or not.
- **Pigs live anywhere.** Pig is not tied to one particular parallel framework.
- **Pigs are domestic animals.** Pig is designed to be easily controlled and modified by its users.
- **Pigs fly.** Pig processes data quickly.

III Pig: What kinds of things can I do with Pig?

Date, temperature

```
2013-06-01,12
2013-06-01,14
2013-06-01,16
2013-06-02,9
2013-06-02,12
2013-06-02,9
...
```

```
-- Load comma-delimited source data
Readings = LOAD '/weather/data.txt' USING PigStorage(',') AS (date:chararray, temp:long);
-- Group the tuples by date
GroupedReadings = GROUP Readings BY date;
-- Get the average temp value for each date grouping
GroupedAvgs = FOREACH GroupedReadings GENERATE group, AVG(Readings.temp) AS avgtemp;
-- Ungroup the dates with the average temp
AvgWeather = FOREACH GroupedAvgs GENERATE FLATTEN(group) as date, avgtemp;
-- Sort the results by date
SortedResults = ORDER AvgWeather BY date ASC;
-- Save the results in the /weather/summary folder
STORE SortedResults INTO '/weather/summary';
```

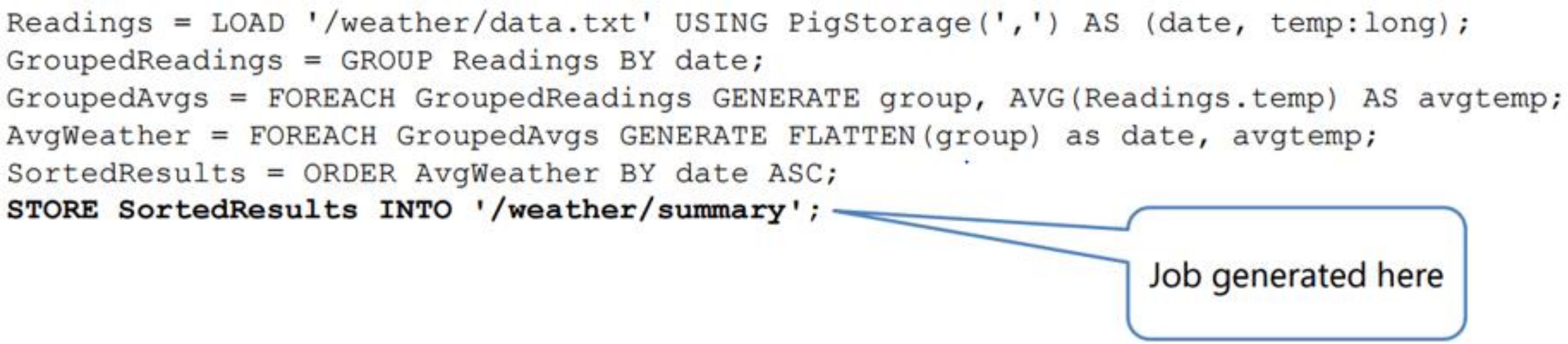
```
2013-06-01  14.00
2013-06-02  10.00
```

Common Pig Latin Operations

- LOAD
- FILTER
- FOR EACH … GENERATE
- ORDER
- JOIN
- GROUP
- FLATTEN
- LIMIT
- DUMP
- STORE

III Pig: What kinds of things can I do with Pig?

1. Pig generates Map and Reduce operations from Pig Latin
2. Jobs are generated on:
   a. DUMP
   b. STORE

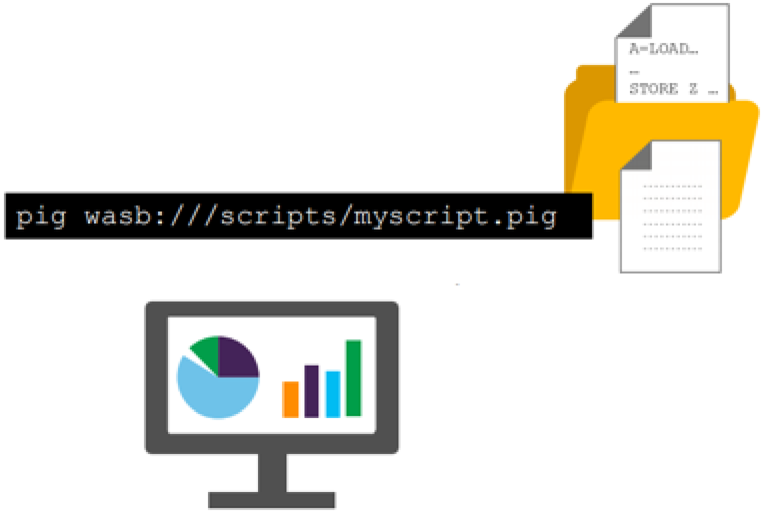


```
Readings = LOAD '/weather/data.txt' USING PigStorage(',') AS (date, temp:long);
GroupedReadings = GROUP Readings BY date;
GroupedAvgs = FOREACH GroupedReadings GENERATE group, AVG(Readings.temp) AS avgtemp;
AvgWeather = FOREACH GroupedAvgs GENERATE FLATTEN(group) as date, avgtemp;
SortedResults = ORDER AvgWeather BY date ASC;
STORE SortedResults INTO '/weather/summary';
```

III Pig: How do I run a Pig script?

1. Save a Pig Latin script file
2. Run the script using Pig

   Pig wasb:///scripts/myscript.pig
3. Consume the results using any Azure storage client
   a. For example, Excel or Power BI
   b. Default output does not include schema – just data

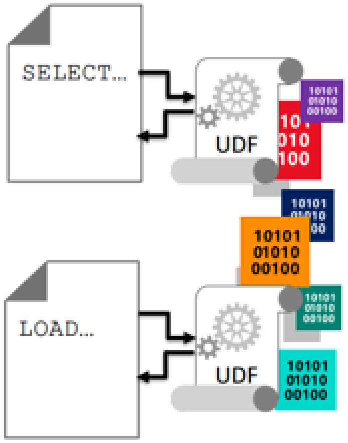


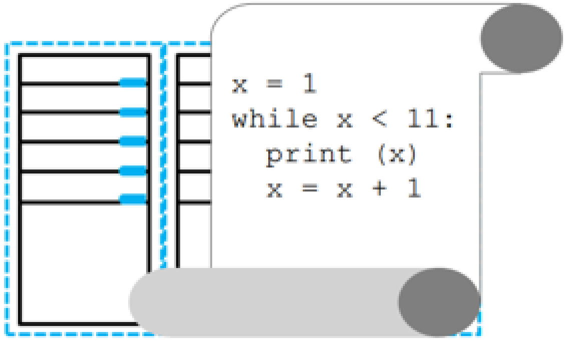
```
pig wasb:///scripts/myscript.pig
```

Demo: Running a Pig Script

III Pig: What are UDFs?

1. User Defined Functions (UDFs) extend the capability of Hive and Pig
2. Simpler than writing custom MapReduce components
3. Can be implemented using many languages, for example:
   a. Java
   b. C#
   c. Python
4. Python is a (relatively) simple scripting language – ideal for UDFs
   a. Intuitive syntax
   b. Dynamic typing
   c. Interpreted execution
5. Python is pre-installed on HDInsight clusters
   a. Python 2.7 supports streaming from Hive
   b. Jypthon (a Java implementation of Python) has native support in Pig

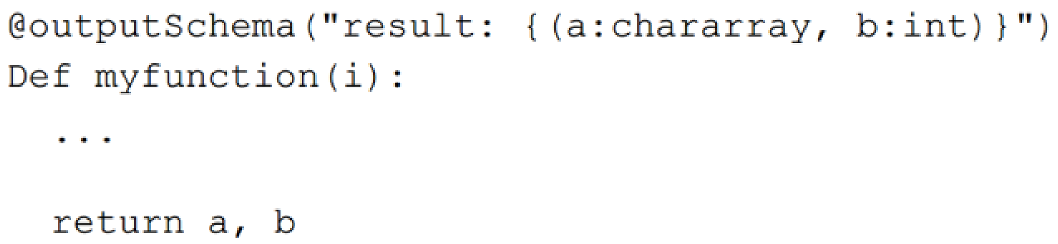




```
x = 1
while x < 11:
   print (x)
   x = x + 1
```

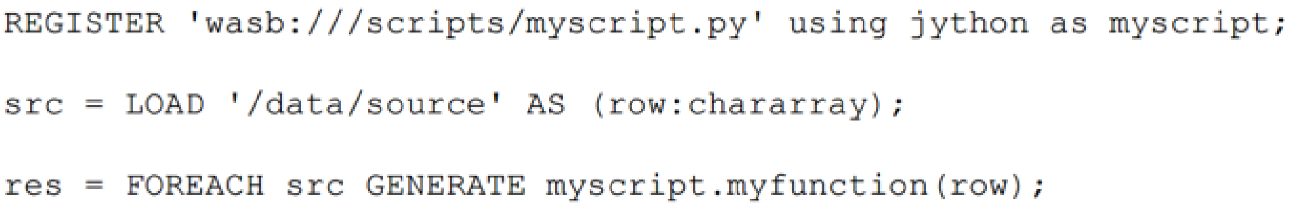
III Pig: How do I use a Python UDF in Pig?

1. Pig natively supports Jython
    a. Define the output schema as a Pig bag
    b. Declare a Python function that receives an input parameter from Pig
    c. Return results as fields based on the output schema



```
@outputSchema("result: {(a:chararray, b:int)}")
Def myfunction(i):
  ...

  return a, b
```

2. Use the Pig FOREACH…GENERATE statement to invoke a UDF

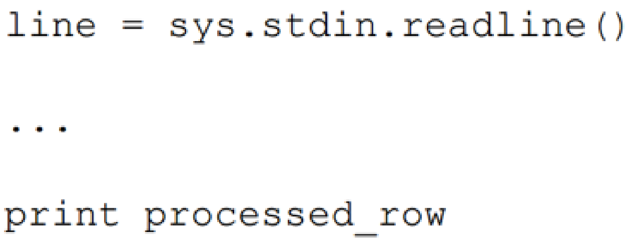


```
REGISTER 'wasb:///scripts/myscript.py' using jython as myscript;

src = LOAD '/data/source' AS (row:chararray);

res = FOREACH src GENERATE myscript.myfunction(row);
```

Demo: Using a Python UDF from Pig

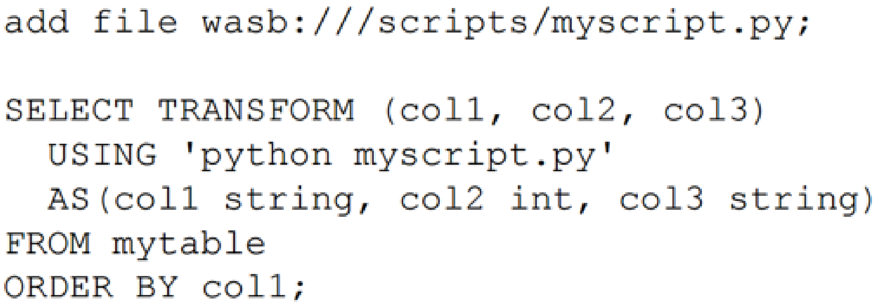
III Pig: How do I use a Python UDF in Hive?

1. Hive exchanges data with Python using a streaming technique
    a. Rows from Hive are passed to Python through STDIN
    b. Processed rows from Python are passed to Hive through STDOUT



```
line = sys.stdin.readline()

...

print processed_row
```

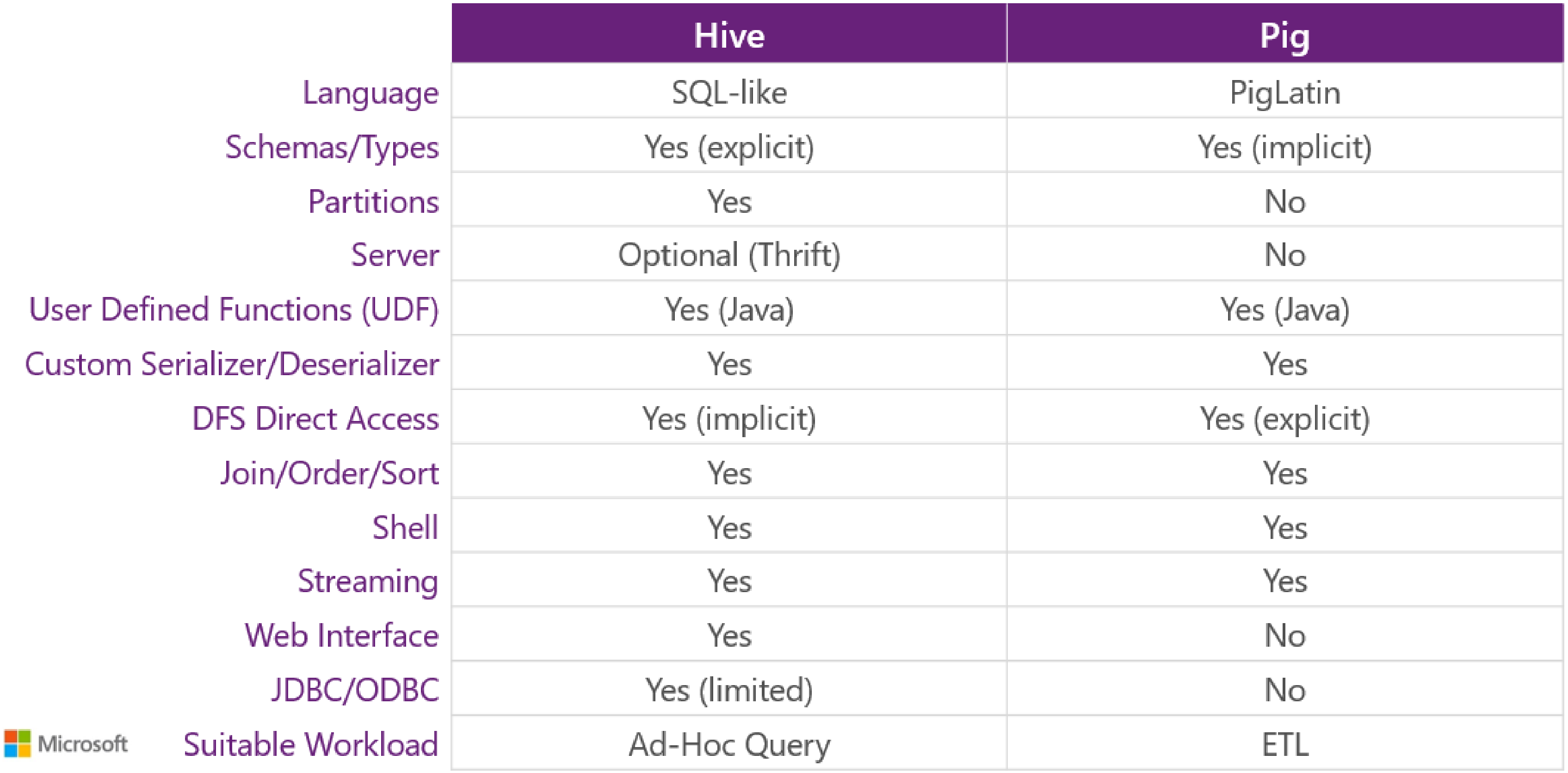
2. Use the Hive TRANSFORM statement to invoke a UDF



```
add file wasb:///scripts/myscript.py;

SELECT TRANSFORM (col1, col2, col3)
  USING 'python myscript.py'
  AS(col1 string, col2 int, col3 string)
FROM mytable
ORDER BY col1;
```

Demo: Using a UDF from Hive

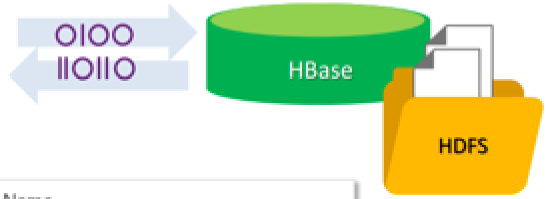
III Hive and Pig – Complementing each other



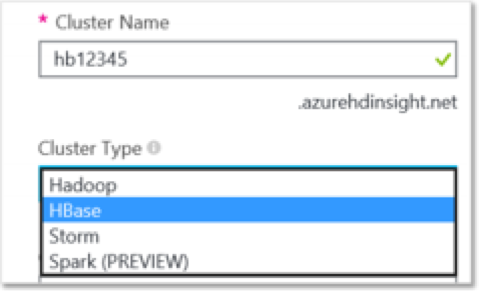
	Hive	Pig
Language	SQL-like	PigLatin
Schemas/Types	Yes (explicit)	Yes (implicit)
Partitions	Yes	No
Server	Optional (Thrift)	No
User Defined Functions (UDF)	Yes (Java)	Yes (Java)
Custom Serializer/Deserializer	Yes	Yes
DFS Direct Access	Yes (implicit)	Yes (explicit)
Join/Order/Sort	Yes	Yes
Shell	Yes	Yes
Streaming	Yes	Yes
Web Interface	Yes	No
JDBC/ODBC	Yes (limited)	No
Suitable Workload	Ad-Hoc Query	ETL

IV HBase: What is Apache HBase?

1. A low-latency, NoSQL database built on Hadoop.
2. Modeled on Google's Big Table.
3. HBase stores data in StoreFiles on HDFS.



1. HDInsight supports an HBase cluster type
   - ✓ Choose Cluster Type in the Azure Portal.
2. Can be provisioned in a virtual network.

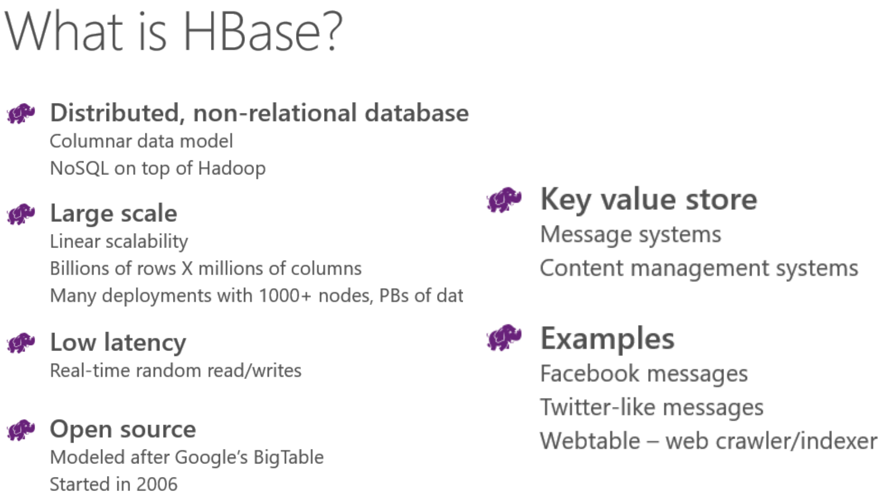


* Cluster Name

hb12345 ✓

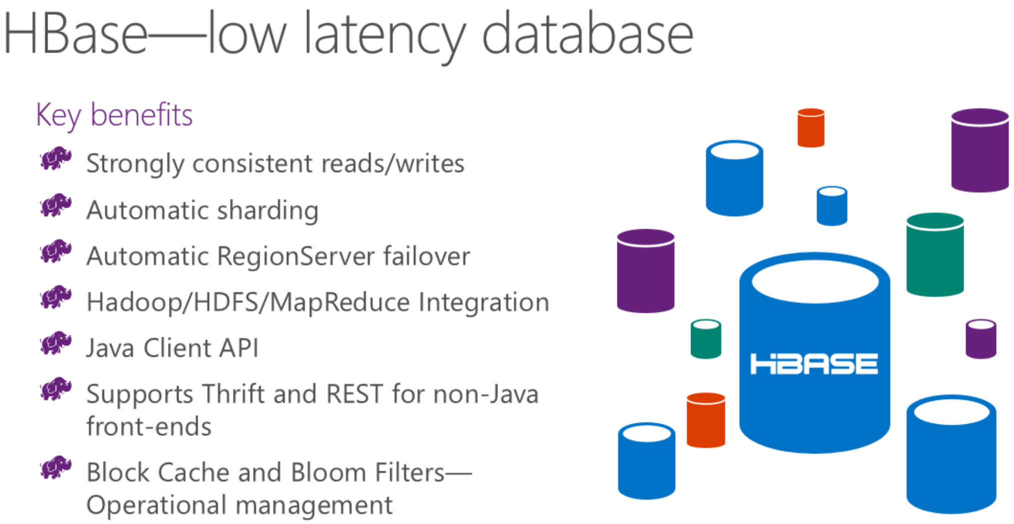
.azurehdinsight.net

Cluster Type



What is HBase?

- **Distributed, non-relational database**
  - Columnar data model
  - NoSQL on top of Hadoop
- **Large scale**
  - Linear scalability
  - Billions of rows X millions of columns
  - Many deployments with 1000+ nodes, PBs of dat
- **Low latency**
  - Real-time random read/writes
- **Open source**
  - Modeled after Google's BigTable
  - Started in 2006
- **Key value store**
  - Message systems
  - Content management systems
- **Examples**
  - Facebook messages
  - Twitter-like messages
  - Webtable – web crawler/indexer



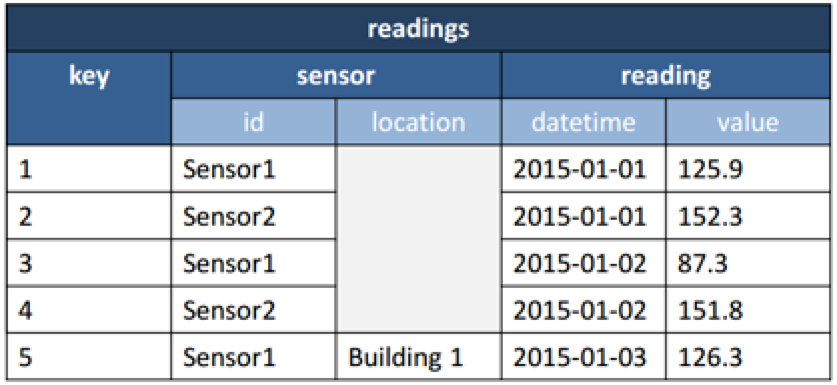
HBase—low latency database

Key benefits

- Strongly consistent reads/writes
- Automatic sharding
- Automatic RegionServer failover
- Hadoop/HDFS/MapReduce Integration
- Java Client API
- Supports Thrift and REST for non-Java front-ends
- Block Cache and Bloom Filters—Operational management

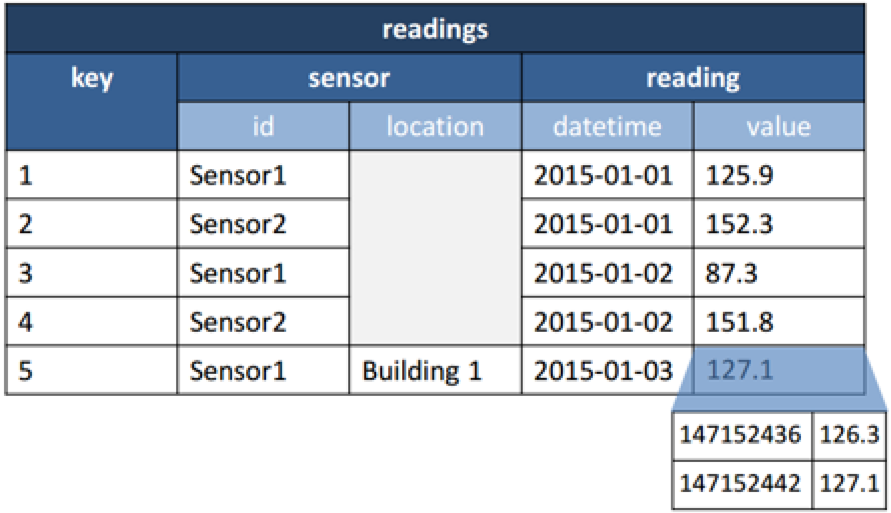
IV HBase: How Does HBase Store Data?

1. Data is stored as key-value pairs.
2. Table schema arranges values into column families.
3. Column family schema is flexible.
4. Columns are row-specific.



readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2		2015-01-01	152.3
3	Sensor1		2015-01-02	87.3
4	Sensor2		2015-01-02	151.8
5	Sensor1	Building 1	2015-01-03	126.3

1. Cells in a table are versioned.
2. Each versioned cell value is indicated by a timestamp.

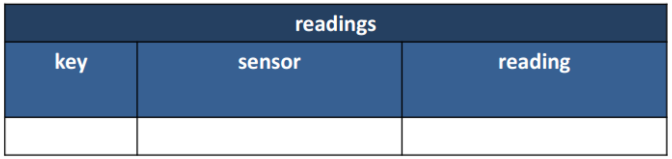


readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2		2015-01-01	152.3
3	Sensor1		2015-01-02	87.3
4	Sensor2		2015-01-02	151.8
5	Sensor1	Building 1	2015-01-03	127.1

147152436	126.3
147152442	127.1

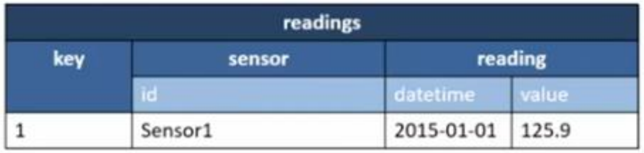
IV HBase: How Do you work with an HBase Table?

```
create 'readings', 'sensor', 'reading'
```



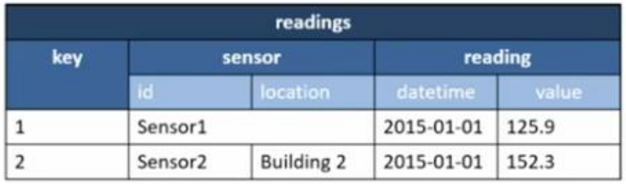
readings		
key	sensor	reading

```
put 'readings', '1', 'reading:value', '125.9'
```



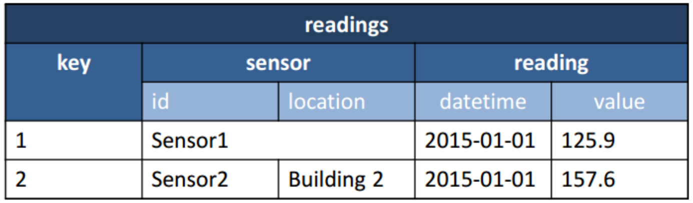
readings			
key	sensor	reading	
	id	datetime	value
1	Sensor1	2015-01-01	125.9

```
put 'readings', '2', 'sensor:location', 'Building 2'
```



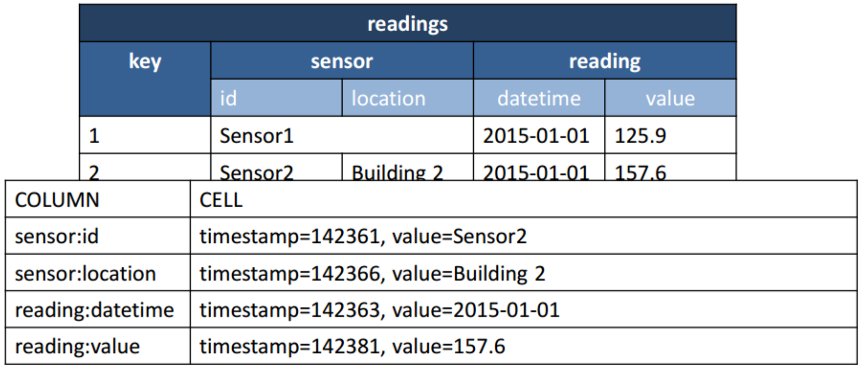
readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	152.3

```
put 'readings', '2', 'reading:value', '157.6'
```



readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6

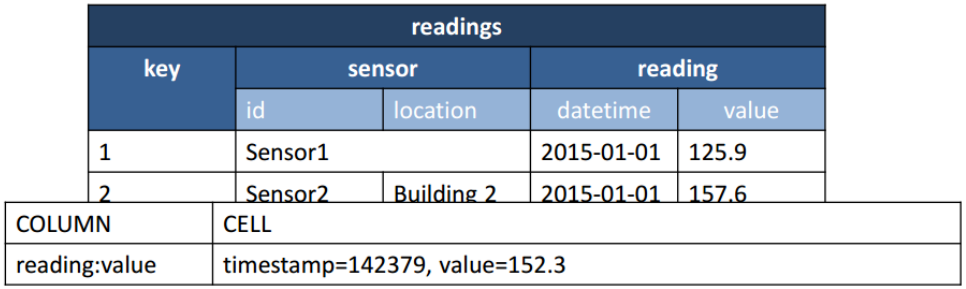
```
get 'readings', '2'
```



readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6

COLUMN	CELL
sensor:id	timestamp=142361, value=Sensor2
sensor:location	timestamp=142366, value=Building 2
reading:datetime	timestamp=142363, value=2015-01-01
reading:value	timestamp=142381, value=157.6

```
get 'readings', '2', {COLUMN => [reading:value]}
```

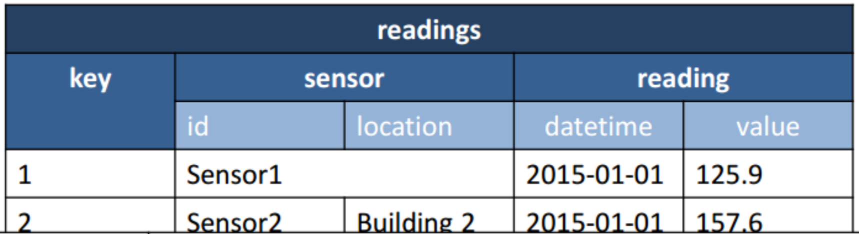


readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6

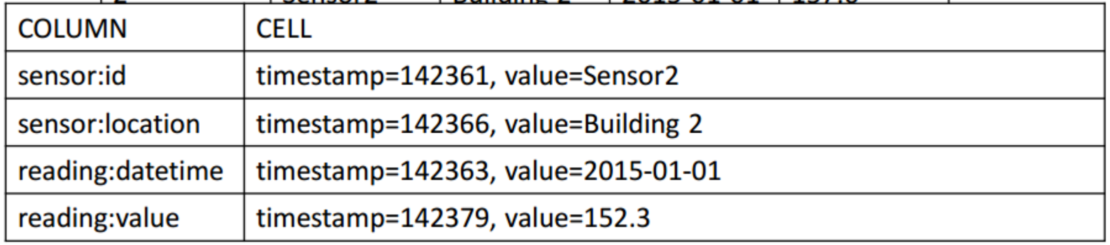
COLUMN	CELL
reading:value	timestamp=142379, value=152.3

IV HBase: How Do you work with an HBase Table?

```
get 'readings', '2', {TIMERANGE => [0,142380]}
```

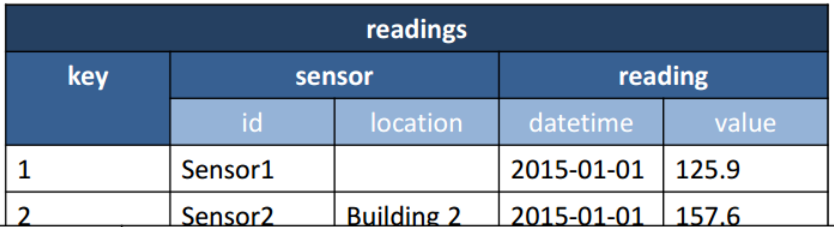


readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6

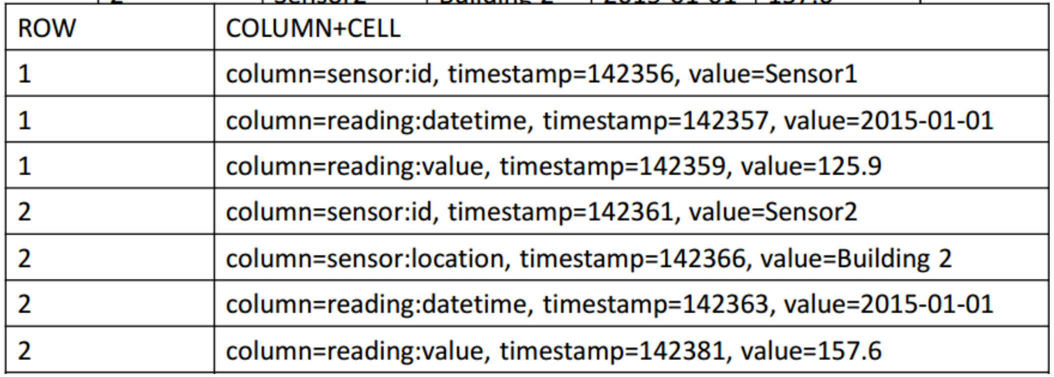


COLUMN	CELL
sensor:id	timestamp=142361, value=Sensor2
sensor:location	timestamp=142366, value=Building 2
reading:datetime	timestamp=142363, value=2015-01-01
reading:value	timestamp=142379, value=152.3

```
scan 'readings'
```



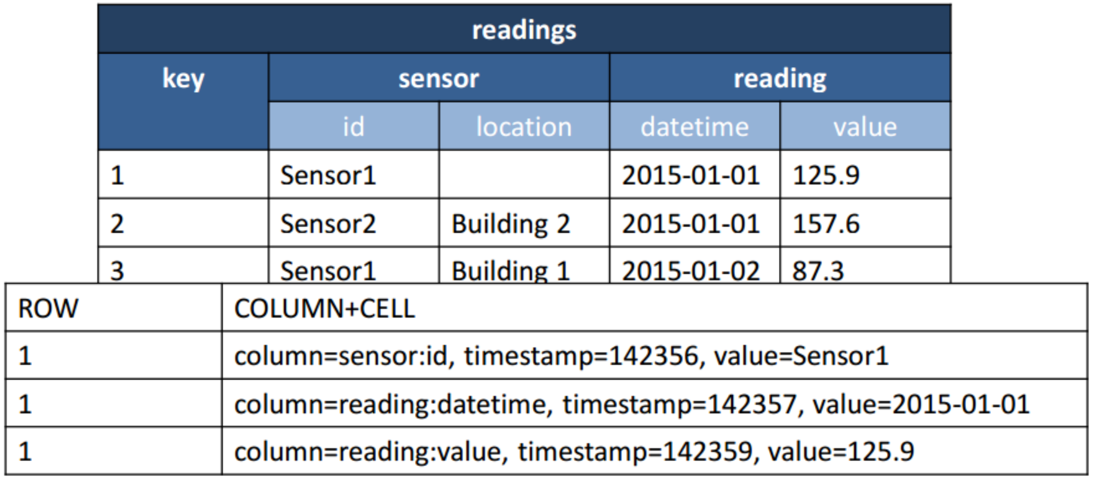
readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6



ROW	COLUMN+CELL
1	column=sensor:id, timestamp=142356, value=Sensor1
1	column=reading:datetime, timestamp=142357, value=2015-01-01
1	column=reading:value, timestamp=142359, value=125.9
2	column=sensor:id, timestamp=142361, value=Sensor2
2	column=sensor:location, timestamp=142366, value=Building 2
2	column=reading:datetime, timestamp=142363, value=2015-01-01
2	column=reading:value, timestamp=142381, value=157.6

IV HBase: How Do you work with an HBase Table?

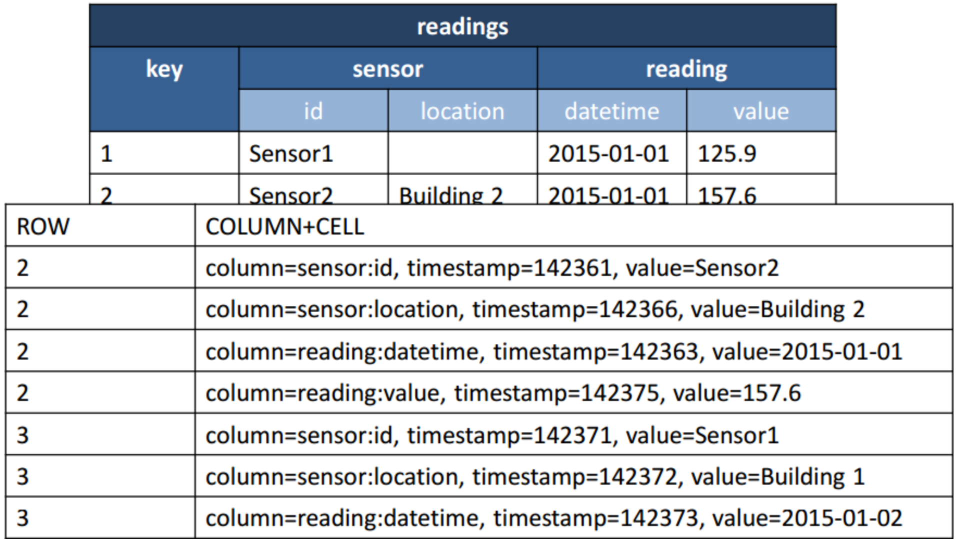
```
scan 'readings', {LIMIT => 1}
```



readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6
3	Sensor1	Building 1	2015-01-02	87.3

ROW	COLUMN+CELL
1	column=sensor:id, timestamp=142356, value=Sensor1
1	column=reading:datetime, timestamp=142357, value=2015-01-01
1	column=reading:value, timestamp=142359, value=125.9

```
scan 'readings', {STARTROW=>'2', STOPROW=>'3'}
```

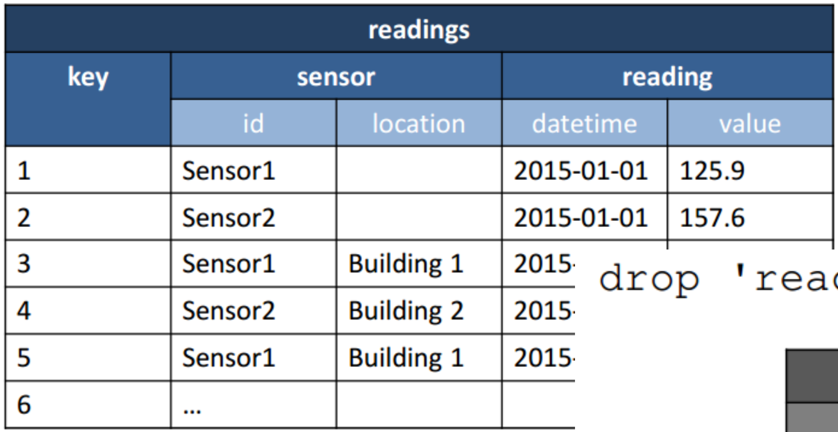


readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2	Building 2	2015-01-01	157.6

ROW	COLUMN+CELL
2	column=sensor:id, timestamp=142361, value=Sensor2
2	column=sensor:location, timestamp=142366, value=Building 2
2	column=reading:datetime, timestamp=142363, value=2015-01-01
2	column=reading:value, timestamp=142375, value=157.6
3	column=sensor:id, timestamp=142371, value=Sensor1
3	column=sensor:location, timestamp=142372, value=Building 1
3	column=reading:datetime, timestamp=142373, value=2015-01-02

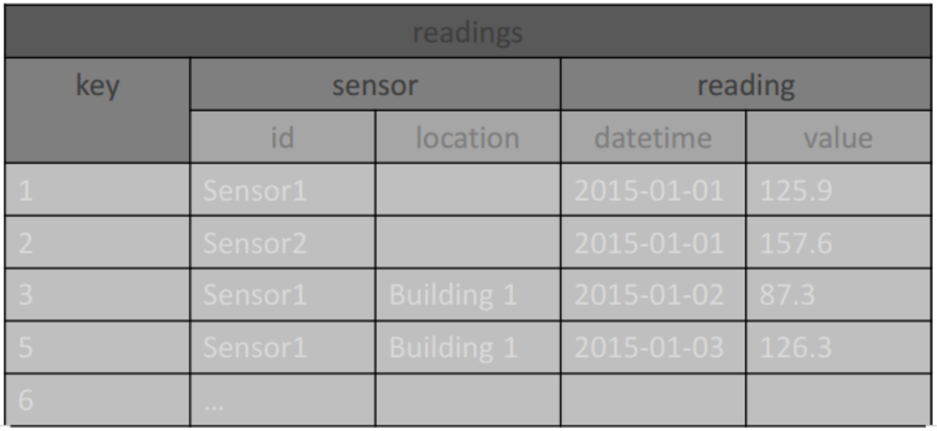
IV HBase: How Do you work with an HBase Table?

```
delete 'readings', '2', 'sensor:location'
```



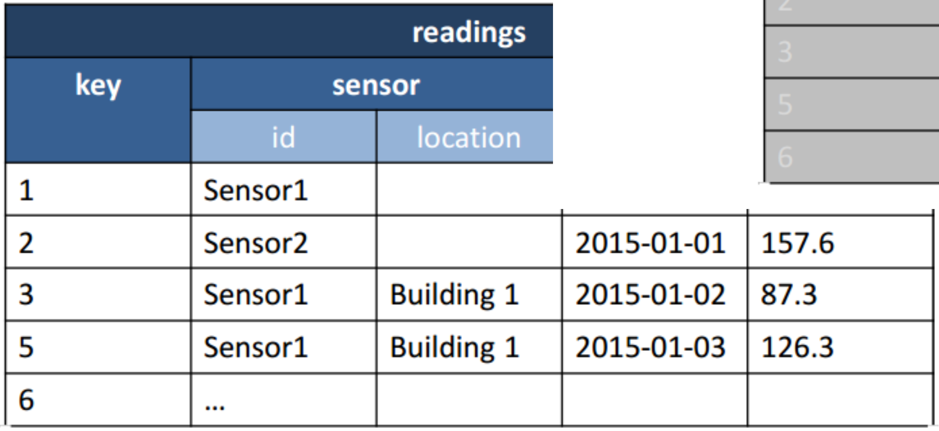
readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2		2015-01-01	157.6
3	Sensor1	Building 1	2015-	
4	Sensor2	Building 2	2015-	
5	Sensor1	Building 1	2015-	
6	…			

```
drop 'readings'
```



readings				
key	sensor		reading	
	id	location	datetime	value
1	Sensor1		2015-01-01	125.9
2	Sensor2		2015-01-01	157.6
3	Sensor1	Building 1	2015-01-02	87.3
5	Sensor1	Building 1	2015-01-03	126.3
6	…			

```
deleteall 'readings', '4'
```

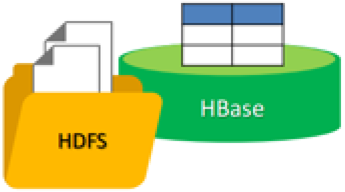


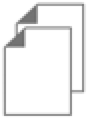
readings				
key	sensor			
	id	location		
1	Sensor1			
2	Sensor2		2015-01-01	157.6
3	Sensor1	Building 1	2015-01-02	87.3
5	Sensor1	Building 1	2015-01-03	126.3
6	…			

IV HBase: How to use HBase with Hive?

How Do you Bulk Load Data into HBase?

1. Upload data to HDFS
   - ✓ In Azure Storage
2. Import into a StoreFile
3. Load the StoreFile to an HBase table

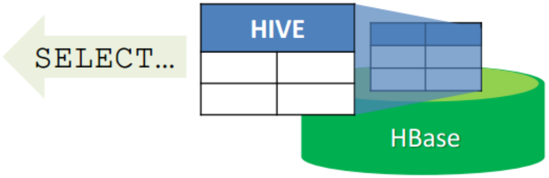




ImportTSV

LoadIncrementalHFiles

How Do You Query HBase Tables from Hive?



```
CREATE EXTERNAL TABLE hivetable
(key STRING, col1 STRING, col2 STRING)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStoragehandler'
WITH SERDEPROPERTIES
('hbase.columns.mapping' = ':key,cf:col1, cf:col2')
TBLPROPERTIES('hbase.table.name' = 'hbtable')
```

IV Sqoop: What is Sqoop?

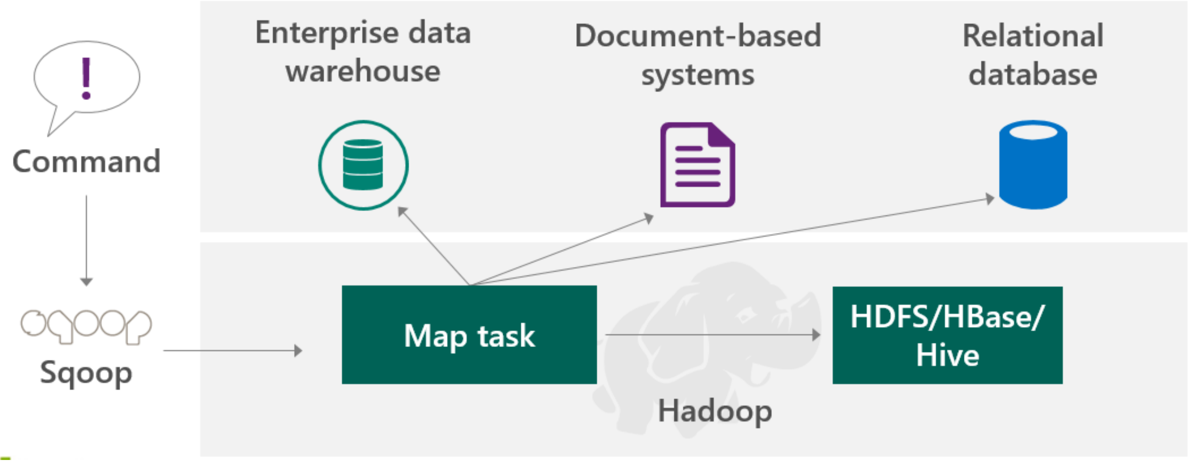
1. Sqoop is a database integration service
   a. Built on open source Hadoop technology
   b. Enables bi-directional data transfer between Hadoop clusters and databases via JDBC

2. Basic syntax

   sqoop command --arg1, --arg2, …, --argN

3. Commands
   a. import / export
   b. help
   c. import-all-tables
   d. create-hive-tables
   e. list-databases / list-tables
   f. eval
   g. codegen
   h. version

Sqoop is designed to efficiently transfer bulk data between Apache Hadoop and structured datastores such as relational databases



IV Sqoop: How do I run Sqoop commands?

1. sqoop import

    --connect jdbc-connection-string

    --username user-name --password password | -P

    --table table-name --columns col,…,colN | --query ‘SELECT…’

    --warehouse-dir | --target-dir path

    --fields-terminated-by char --lines-terminated-by char

    --hive-import [--hive-overwrite]

    -m | --num-mappers number-of-mappers

2. sqoop export

    --connect jdbc-connection-string

    --username user-name --password password | -P

    --table table-name

    --export-dir path

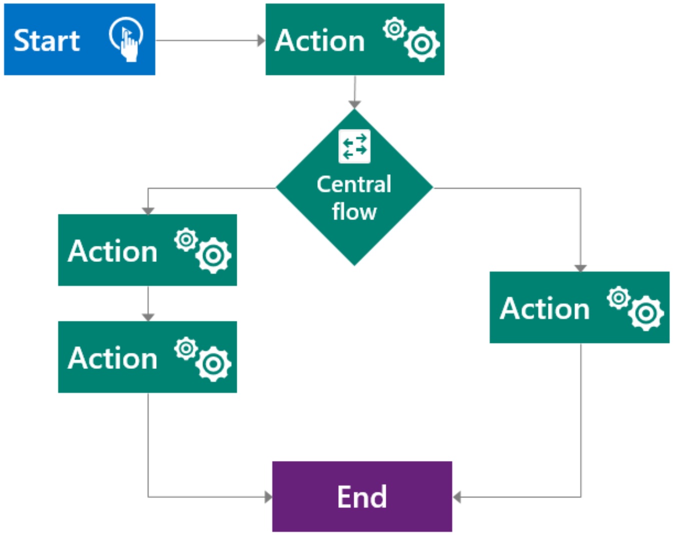
    --fields-terminated-by char --lines-terminated-by char

    -m | --num-mappers number-of-mappers

IV: Oozie and Mahout

Oozie workflow

- Consists of workflow.xml file and the necessary files for the workflow itself
- Workflow saved to WASB store
- Control flow nodes for determining the execution path
- Action nodes for execution the job or tasks



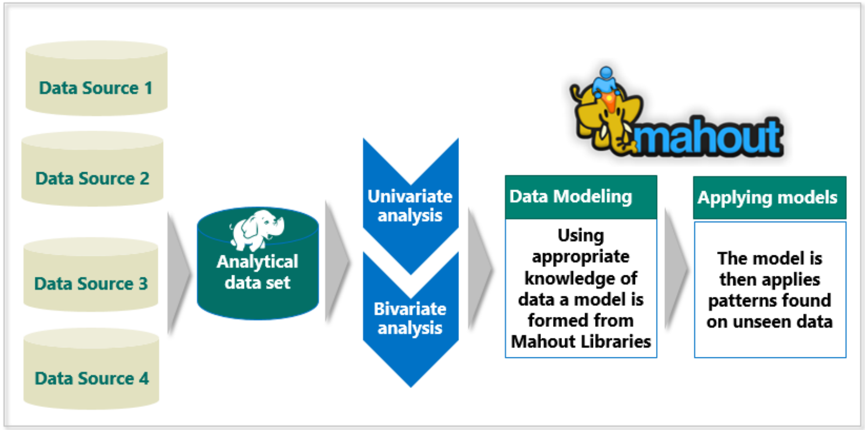
Mahout

Mahout is an Apache project to implement machine learning algorithms in Hadoop

Uses Hadoop to distribute algorithms to parallelize large scale workloads

Three different categories of algorithms

- **Recommendation:** Uses user information and community information to build recommendation. E.g. Netflix, Amazon, Pandora
- **Classification:** Uses known data to classify new data into known buckets. E.g. Spam Detection
- **Clustering:** Forms groups of data into similar categories. E.g. Bing News



Q and A