Microsoft

AZURE HDINSIGHT

ANIRUDH GUPTA

Intern, CDP PARTNERS TEAM

INTRODUCTION

Data Really Powers Everything That We Do

Microsoft Azure HDInsight is Microsoft's **100** percent compliant distribution of Apache Hadoop on Microsoft Azure. It is basically a **Platform as a service(PaaS)** offered by Microsoft to its customers. With no tension about the maintenance of the infrastructure, one can easily use the Hadoop technologies using Azure HDInsight. One the great advantage of this Microsoft offering is high availability and reliability of the infrastructure with low cost. All the standard Hadoop concepts and technologies apply in HDInsight, so learning the Hadoop stack helps to learn the HDInsight service.

Azure HDInsight deploys and provisions managed Apache Hadoop clusters in the cloud, providing a software framework designed to process, analyze, and report on big data with high reliability and availability. HDInsight uses the Hortonworks Data Platform (HDP) Hadoop distribution. Hadoop often refers to the entire Hadoop ecosystem of components, which includes Apache HBase, Apache Spark, and Apache Storm, as well as other technologies under the Hadoop umbrella. All these components are part of HDInsight too.

In Introducing Microsoft Azure HDInsight, we cover what big data really means, how we can use it to our advantage in any organization, how we can use this Hadoop services in HDInsight. Before starting with an overview of big data and Hadoop, let's have a look at the advantages of HDInsight.

- Quickly deploy the system from a portal or through Windows PowerShell scripting, without having to create any physical or virtual machines.
- Implement a small or large number of nodes in a cluster.
- Pay only for what you use.
- When the job is complete, deprovision the cluster and, of course, stop paying for it.
- Use Microsoft Azure Storage so that even when the cluster is deprovisioned, the data can be retained
- The HDInsight service works with input-output technologies from Microsoft or other vendor.

For more information: https://azure.microsoft.com/en-in/documentation/articles/hdinsight-hadoop-introduction/

WHAT IS BIG DATA?

Big Data: Techniques and Technologies That Make Handling Data at Extreme Scale Economical.

Big data refers to data being collected in ever-escalating volumes, at increasingly higher velocities, and in an expanding variety of unstructured formats and variable semantic contexts. Big data describes any large body of digital information, from the text in a Twitter feed, to the sensor information from industrial equipment, to information about customer browsing and purchases on an online catalog. Big data can be historical (meaning stored data) or real-time (meaning streamed directly from the source).

For big data to provide actionable intelligence or insight, not only must you collect relevant data and ask the right questions, but also the data must be accessible, cleaned, analyzed, and then presented in a useful way. That's where big data analysis on Hadoop in HDInsight can help.

Big data is commonly defined in terms of exploding data volume, increases in data variety, and rapidly increasing data velocity

• **Volume:** The data exceeds the physical limits of vertical scalability, implying a scale-out solution.

• **Velocity**: Data streams in at an unprecedented speed and must be dealt with in a timely manner.

• **Variety:** Many different formats of data make the integration difficult and expensive.

• **Variability:** In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks

Differentiating between Big Data and Traditional Enterprise Relational Data

Thinking of Big Data as "just lots more enterprise data" is tempting, but it's a serious mistake. First, Big Data is notably larger — often by several orders of magnitude. Secondly, Big Data is commonly generated outside of traditional enterprise applications. And finally, Big Data is often composed of unstructured or semi-structured information types that continually arrive in enormous amounts.

Some Facts

- Every day 2.5 quintillion bytes or 2.5 exabytes are generated, that number is estimated to double every 40 month
- Twitter: 12 terabytes of Tweets every day
- It is estimated that Walmart collects more than 2.5 petabytes of data every hour from its customer transactions. A petabyte is one quadrillion bytes, or the equivalent of about 20 million filing cabinets' worth of text.
- Bad data or poor quality data costs US businesses $600 billion annually.

HADOOP

Apache Hadoop: Born Out of a Need to Process an Avalanche of Big Data.

Apache Hadoop, an open-source software framework, is one way of solving a big data problem by using a scale-out "divide and conquer" approach. The grocery store analogy works quite well here because the two problems in big data (moving the processing to the data and then combining it all again) are solved with two components that Hadoop uses: Hadoop Distributed File System (HDFS) and MapReduce. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

2002	Doug Cutting and Michael J. Cafarella developed a tool to support distribution for the Nutch search engine project.
2003	Google File System paper released
2004	Google Released MapReduce paper
2006	Hadoop is born from Nutch 197 at Yahoo
2008	Hadoop becomes a Top-level Apache project
2009	First Commercial Hadoop Distribution
2010	Hive, Pig, HBase graduate

HADOOP CLUSTER

A cluster is a set of connected computers working together in a way that they could be viewed as a single system. Clusters are basically used for parallel processing, load balancing and fault tolerance. A Hadoop cluster is a special type of computational cluster designed specifically for storing and analyzing huge amounts of data in a distributed computing environment. Adding more systems increases the power of cluster.

A Hadoop cluster has a **Master Node** and many **Slave Nodes.** Each node in the cluster consists of both the HDFS component and MapReduce component.

- **Master Node**: This node manages all the slave machines and is responsible for processing data on the slave machines. It consists of NameNode (HDFS component) and JobTracker (MapReduce component).
- **Salve Node**: The slave nodes work according to the instructions given by the Master Node and process the data accordingly. They consist of DataNode (HDFS component) and Task Tracker (MapReduce component).

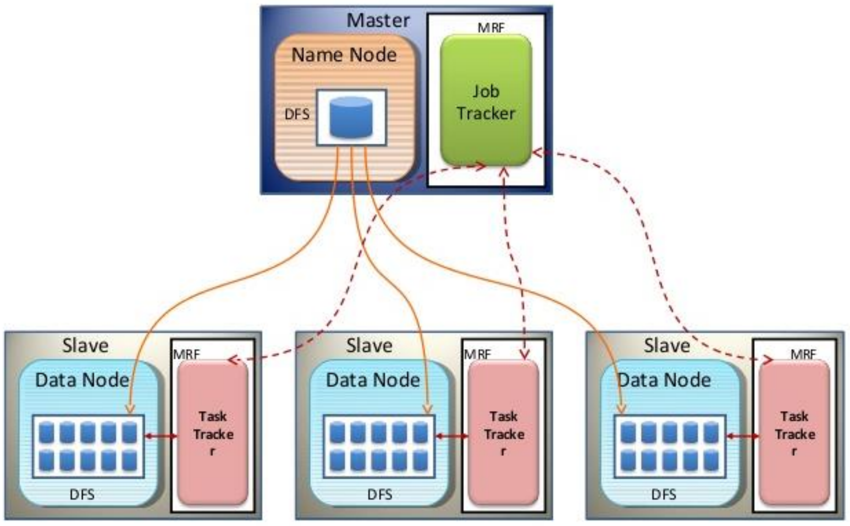


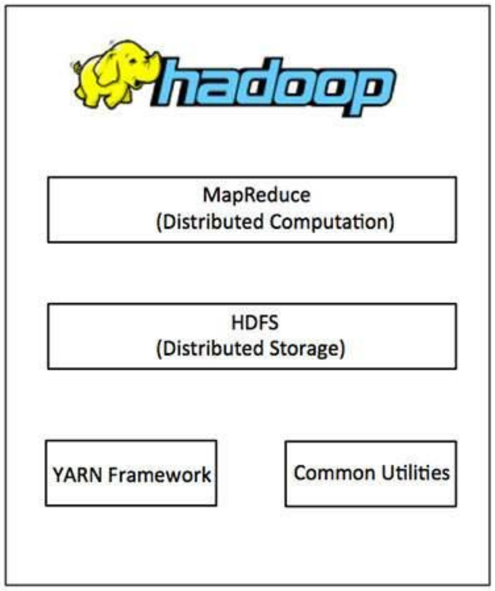
Fig: Hadoop Cluster

Link: http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

HADOOP ARCHITECTURE

Hadoop framework basically includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provide filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.



Some Facts

The name Hadoop is not an acronym; it's a made-up name. The project's creator, Doug Cutting, explains how the name came about:

> *The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such.*

Subprojects and "contrib" modules in Hadoop also tend to have names that are unrelated to their function, often with an elephant or other animal theme ("Pig," for example).

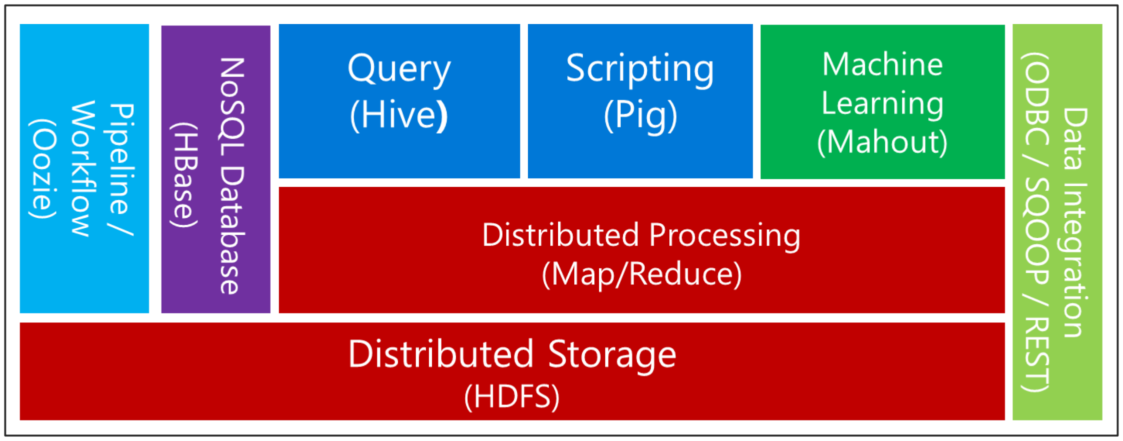
HADOOP ECOSYSTEM ON HDINSIGHT

- **Data Management**– Store and process vast quantities of data in a storage layer that scales linearly. Hadoop Distributed File System (HDFS) is the core technology for the efficient scale out storage layer, and is designed to run across low-cost commodity hardware. Apache Hadoop YARN is the pre-requisite for Enterprise Hadoop as it provides the resource management and pluggable architecture for enabling a wide variety of data access methods to operate on data stored in Hadoop with predictable performance and service levels.
    - **HDFS–** Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers.
- **Data Access**– Interact with your data in a wide variety of ways – from batch to real-time.
    - **Apache Hive–** It is the most widely adopted Data Access Technology. Built on the MapReduce framework, Hive is a data warehouse that enables easy data summarization and ad-hoc queries via an SQL-like interface for large datasets stored in HDFS.
    - **Apache Pig–** A platform for processing and analyzing large data sets. Pig consists of a high-level language (Pig Latin) for expressing data analysis programs paired with the MapReduce framework for processing these programs.
    - **MapReduce–** MapReduce is a framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable and fault-tolerant manner.
    - **Apache Spark–** It is ideal for in-memory data processing. It allows data scientists to implement fast, iterative algorithms for advanced analytics such as clustering and classification of datasets.
    - **Apache Storm–** It is a distributed real-time computation system for processing fast, large streams of data adding reliable real-time data processing capabilities to Apache Hadoop® 2.x
    - **Apache HBase–** A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications.
    - **Apache Tez–** It generalizes the MapReduce paradigm to a more powerful framework for executing a complex DAG (directed acyclic graph) of tasks for near real-time big data processing.
    - **Apache HCatalog–** A table and metadata management service that provides a centralized way for data processing systems to understand the structure and location of the data stored within Apache Hadoop.

- **Apache Mahout–** Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering.

- **Data Governance and Integration**– Quickly and easily load data, and manage according to policy.
    - **Apache Sqoop–** It is a tool that speeds and eases movement of data in and out of Hadoop. It provides a reliable parallel load for various, popular enterprise data sources.

- **Operations**– Provision, manage, monitor and operate Hadoop clusters at scale.
    - **Apache Ambari–** An open source installation lifecycle management, administration and monitoring system for Apache Hadoop clusters.
    - **Apache Oozie–** Oozie Java Web application used to schedule Apache Hadoop jobs. Oozie combines multiple jobs sequentially into one logical unit of work.
    - **Apache ZooKeeper–** A highly available system for coordinating distributed processes. Distributed applications use ZooKeeper to store and mediate updates to important configuration information.



Pipeline / Workflow (Oozie)
NoSQL Database (HBase)
Query (Hive)
Scripting (Pig)
Machine Learning (Mahout)
Distributed Processing (Map/Reduce)
Distributed Storage (HDFS)
Data Integration (ODBC / SQOOP / REST)

Fig: Hadoop Architecture

Links:

- http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/#section_2
- http://hadoopilluminated.com/hadoop_illuminated/Intro_To_Hadoop.html

HDFS-Hadoop Distributed File System

Apache Hadoop: Born Out of a Need to Process an Avalanche of Big Data.

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and written entirely in Java. Google provided only a white paper, without any implementation; however, around 90 percent of the GFS architecture has been applied in its implementation in the form of HDFS.

HDFS is a highly scalable, distributed, load-balanced, portable, and fault-tolerant (with built-in redundancy at the software level) storage component of Hadoop. In other words, HDFS is the underpinnings of the Hadoop cluster. It provides a distributed, fault-tolerant storage layer for storing Big Data in a traditional, hierarchical file organization of directories and files. HDFS has been designed to run on commodity hardware.

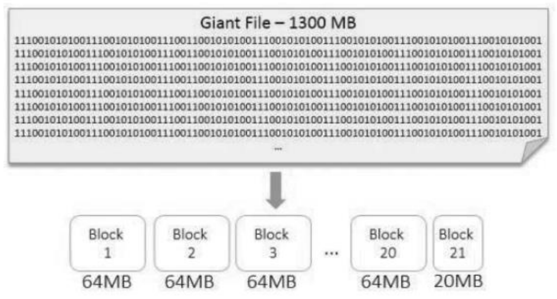
HDFS was originally built as a storage infrastructure for the Apache Nutch web search engine project. It was initially called the Nutch Distributed File System (NDFS).

HDFS ARCHITECTURE

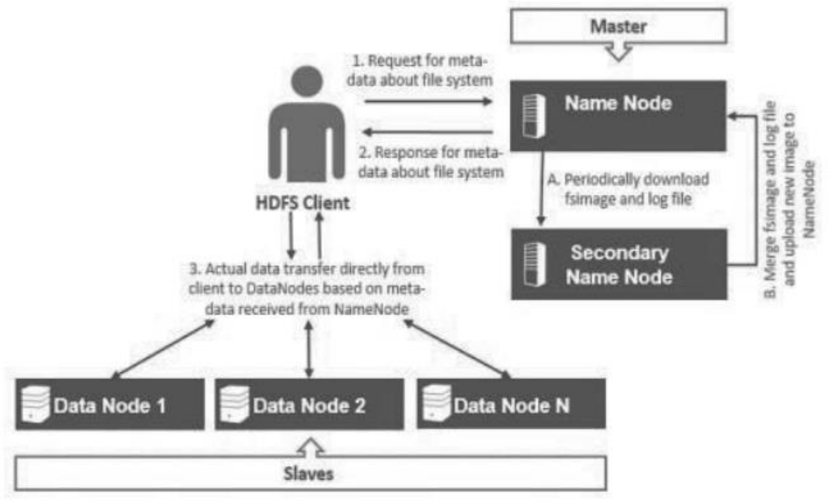
HDFS has a master and slave's architecture in which the master is called the ***name node*** and slaves are called ***data nodes***. An HDFS cluster consists of a single name node that manages the file system namespace (or metadata) and controls access to the files by the client applications, and multiple data nodes (in hundreds or thousands) where each data node manages file storage and storage device attached to it.

- **Data Node:** While Storing a file, HDFS splits the file into one or more blocks (chunks of 64 MB by default). These blocks are then stored in a set of slave nodes called the data nodes, to ensure parallel read and write in a single file. The size of the blocks can be configured by the user at both the cluster level and the file level.

  Multiple copies of blocks are stored across the slave nodes to ensure that the platform is fault tolerant. The multiple copies depend on the replication factor (default is 3 i.e. each block is stored on 3 slave nodes by default). The replication factor can be configured by the user on cluster level, during file creation or at a later stage.



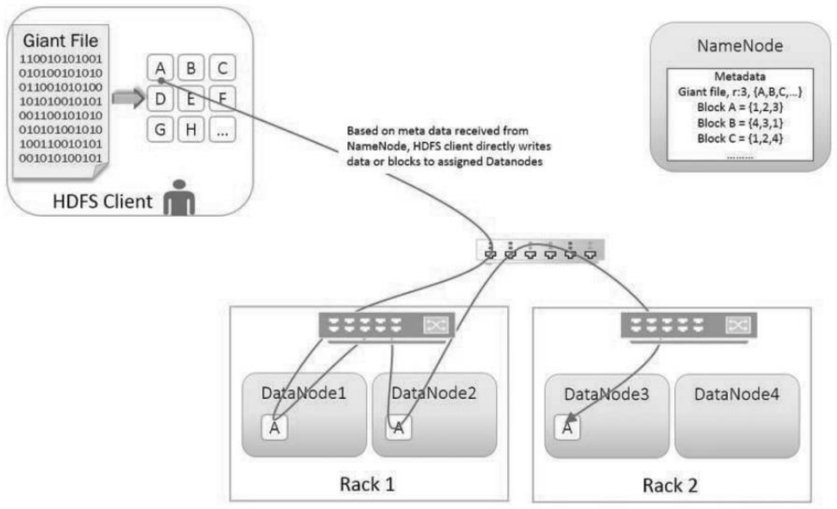
- **Name Node:** The name node consists of the information related to the replication factor of a file. It stores the details of the block location along the information about its copies. The name node is also responsible for managing file system namespace operations, including opening, closing, and renaming files and directories. It records any changes to the file system namespace or its properties.


Master
1. Request for meta-data about file system
2. Response for meta-data about file system
HDFS Client
Name Node
A. Periodically download fsimage and log file
Secondary Name Node
B. Merge fsimage and log file and upload new image to NameNode
3. Actual data transfer directly from client to DataNodes based on meta-data received from NameNode
Data Node 1
Data Node 2
Data Node N
Slaves

WRITING TO HDFS

When a client or any application wants to write a file to HDFS, it reaches out to the NameNode with the details of the file. The Name Node then responds back with the details such as number of blocks of the file, block size, the replication factor, the location where each blocks needs to be stored (on the Data Nodes).

Based on the information received by the Name Node, the client or the application splits the file into blocks of the given size and replicates as per the replication factor.



Let's perform this activity using the figure above. The Block A of file needs to be stored at DataNode1, DataNode2, DataNode3 as per the information given by the NameNode. The client sends the block to the first data node (i.e. DataNode1). After receiving the block, DataNode1 copies the block to the other data node (DataNode2), which then copies it to DataNode3. This helps in saving the time of client as it can start sending the other Block.

When all the instructed data nodes receive the block, each one of them writes a confirmation to the NameNode. After this the first data node (i.e. DataNode1) writes a confirmation to the client.

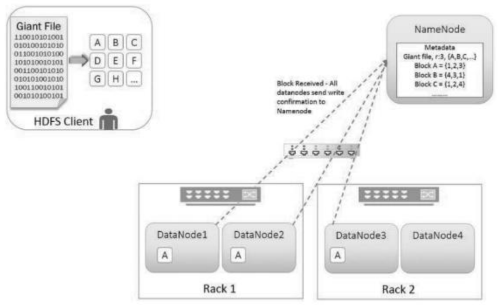


Fig: Data Nodes update the NameNode on receiving the block

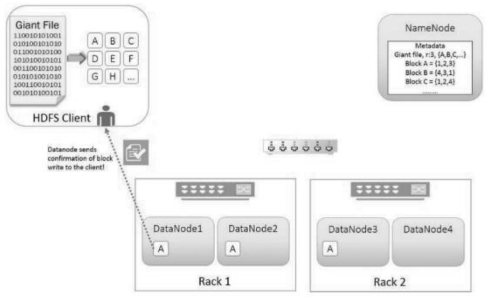


Fig: First node sends acknowledgement to the client.

For better performance, the data nodes do not wait for the whole block to arrive before copying it to the other node. Instead the data transfer from client to the first data node happens in smaller chunks of 4 KB. When DataNode1 receives a chunk, it stores it in its local repository and immediately starts copying it to the other data node. Likewise, the DataNode2 stores and transfers it immediately.

Some Facts

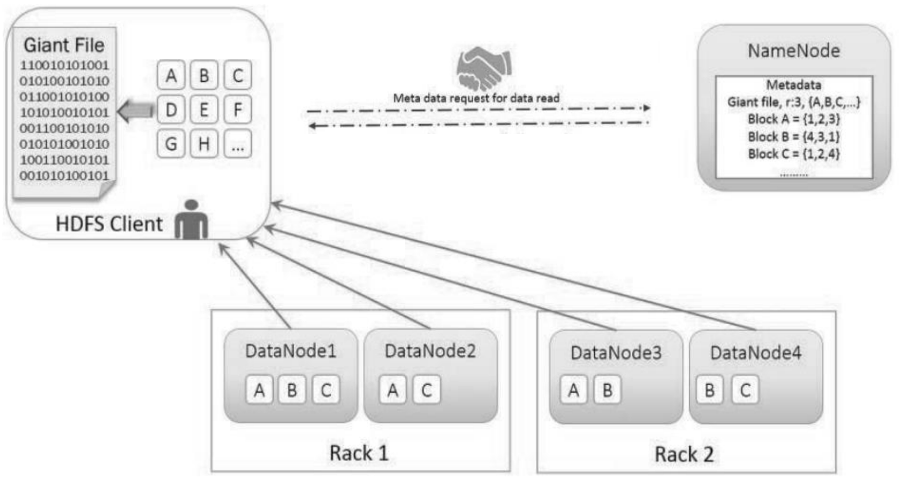
- One of the major advantage of Hadoop, Fault Tolerant, is achieved by the replicated block copies. Whenever any node fails the Name Node gives the information about its copy to do the processing.
- What happens if the NameNode fails?

  The NameNode failure is the recognized single point of failure in Hadoop. This failure existed till Hadoop 2. HDInsight solves this failure by keeping a ***Secondary NameNode*** *with gets updated to the changes in NameNode.*

READING FROM HDFS

To read the file from the HDFS, the client or application contacts the NameNode with the name and location of the file. The NameNode responds back with the information about the various blocks of the files along the Data Nodes where the blocks the stored.

The client now directly contacts the DataNode for the transfer of block (without involving the NameNode) and reads the blocks in parallel. After the transfer of blocks it combines them to form a single file.



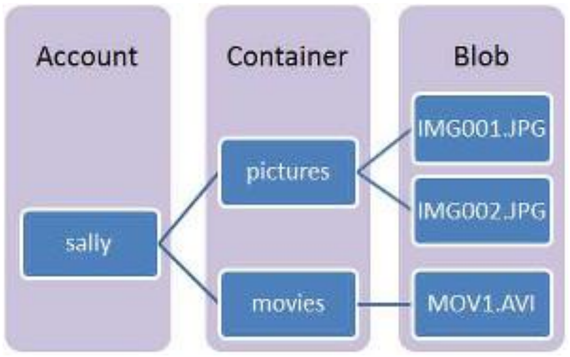
STORAGE ON HDINSIGHT

HDInsight uses Azure Blob Storage as its primary storage account. Azure Blob Storage is a service offering from Microsoft for storing large amount of data (Structured as well as unstructured) which can accessed anywhere in the world. It stores files in the form of blobs. BLOB's stands for Binary Large Objects which are contained in a container.

The main advantage of using the Blob Storage is that it enables deletion of HDInsight clusters that are used for computation without losing user data. The account can be relinked with the cluster when some computation needs to be performed. This is a cost-efficient tool which enables HDInsight to be "**Pay-When-You-Use**" service.

Hadoop uses HDFS to store data but HDInsight stores the data in form of Blobs, then how do the Hadoop services (which are developed to use HDFS) get compatible with HDInsight?

All data resides in Blob Store, whenever any query comes the data(blob) is transferred to Nodes of Cluster for processing. The azure data centre makes this process quite fast and is generally same or better than the data on local disk. Since blobs can't be understood by the Hadoop, a tool named **WebHDFS** comes into picture. WebHDFS is used to make HTTP calls. It helps an external application in accessing and managing the files in HDFS. It also acts as a wrapper over blobs and provides data to the Hadoop services in their desired format.



For more information, visit:

- https://hadoop.apache.org/docs/stable2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html
- http://hadoopilluminated.com/hadoop_illuminated/HDFS_Intro.html

Some Facts

- Thinking about how Hadoop knows about the node failure??
  Well, Hadoop is smart. The slaves send regular heartbeats to the master node and if the slave is silent over a period of time (say 10 mins) the master assigns the task to some other slave node.
- A storage account can store up to 500 TB of Data and each Azure Subscription can have 100 such storage accounts. Also, multiple storage accounts can be linked to HDInsight now.

MAP REDUCE

The key for Hadoop processing

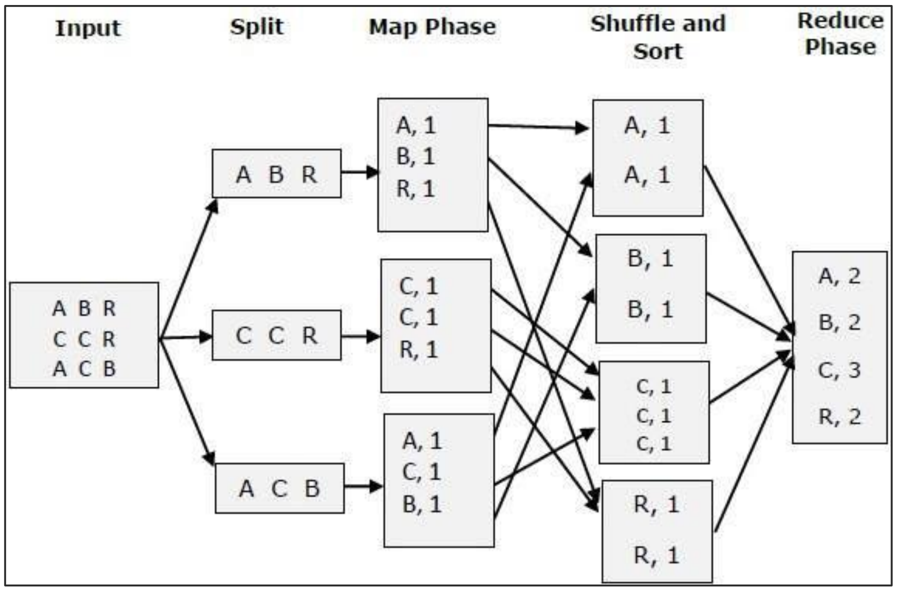
MapReduce is the key algorithm that the Hadoop data processing engine uses to distribute the work around the cluster. MapReduce is a software framework for processing large datasets in a distributed fashion over a several machines. MapReduce divides a task into small parts and organizes them into key value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with greater reliability. The core idea behind MapReduce is mapping your data set into a collection of pairs, and then reducing overall pairs with the same key.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The **Map** task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The **Reduce** task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job.

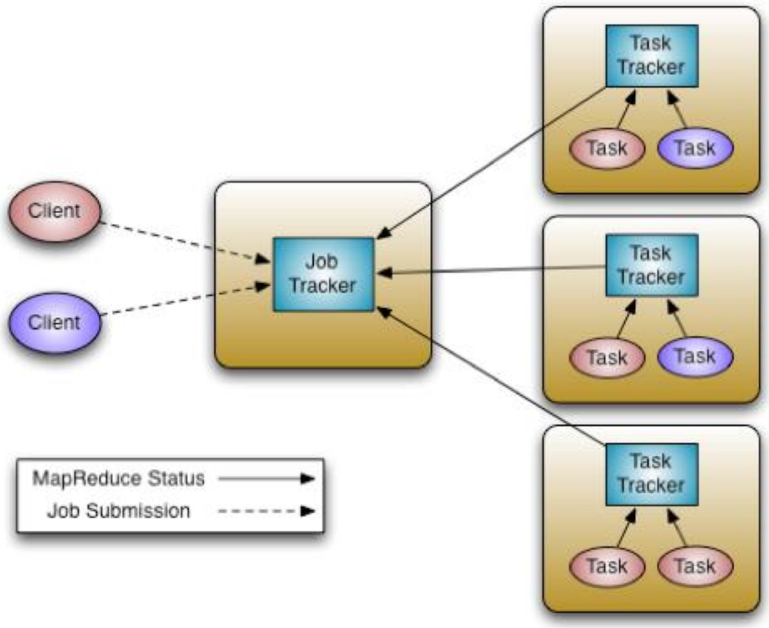
Below is a figure that explains the working of mapreduce job. The job is simple, to find frequency of letters. The map function receives the input file blocks and map the letter with its frequency (key value pair). Then the shuffle and sort phase occurs which combines the key-value pairs having same keys and also sorts them on the basis of their key values. These are now transferred to the reducer phase where the frequency corresponding to the key is calculated and given as the output.



There are 2 main components which help in the working of MapReduce Jobs. They are described below:

- **Job Tracker** is the master of the system which manages the jobs and resources in the cluster (TaskTrackers). The JobTracker tries to schedule each map as close to the actual data being processed i.e. on the TaskTracker which is running on the same DataNode as the underlying block. The Job-tracker creates an execution plan and is responsible for coordination and fault tolerance. It takes help of NameNode for getting information about the nodes.

- **Task Trackers** are the slaves which are deployed on each machine. They are responsible for running the map and reduce tasks as instructed by the JobTracker.



For more information, visit:

- http://hadoopilluminated.com/hadoop_illuminated/MapReduce_Intro.html
- http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/#concepts-mapreduce-yarn

CREATING A CLUSTER

A Hadoop cluster consists of several virtual machines (nodes) which are used for distributed processing of tasks on the cluster. Azure abstracts the implementation details of installation and configuration of individual nodes, so you only have to provide general configuration information.

CLUSTER TYPES

Cluster type	Use this if you need...
Hadoop	Query and analysis (batch jobs).
HBase	NoSQL data storage.
Storm	Real-time event processing.
Spark (Preview)	In-memory processing, interactive queries, micro-batch stream processing.

The following are the basic configuration options for creating an HDInsight cluster.

- **Cluster Name:** Cluster name is used to identify a cluster. Cluster name must be globally unique, and it must follow the following naming guidelines:
    - The field must be a string that contains between 3 and 63 characters
    - The field can contain only letters, numbers, and hyphens.

- **Subscription name:** This field is to specify the Azure Subscription
    - Each HDInsight cluster is tied to one Azure subscription.

- **Cluster type:** This field is to choose the type and OS of the nodes in the cluster
    - **Operating system:** HDInsight clusters can be created on one of the following two operating systems:
        - HDInsight on Linux (Ubuntu 12.04 LTS for Linux)
        - HDInsight on Windows (Windows Server 2012 R2 Datacenter)
    - **HDInsight version:** It is used to determine the version of Hadoop to use for the cluster in HDInsight.

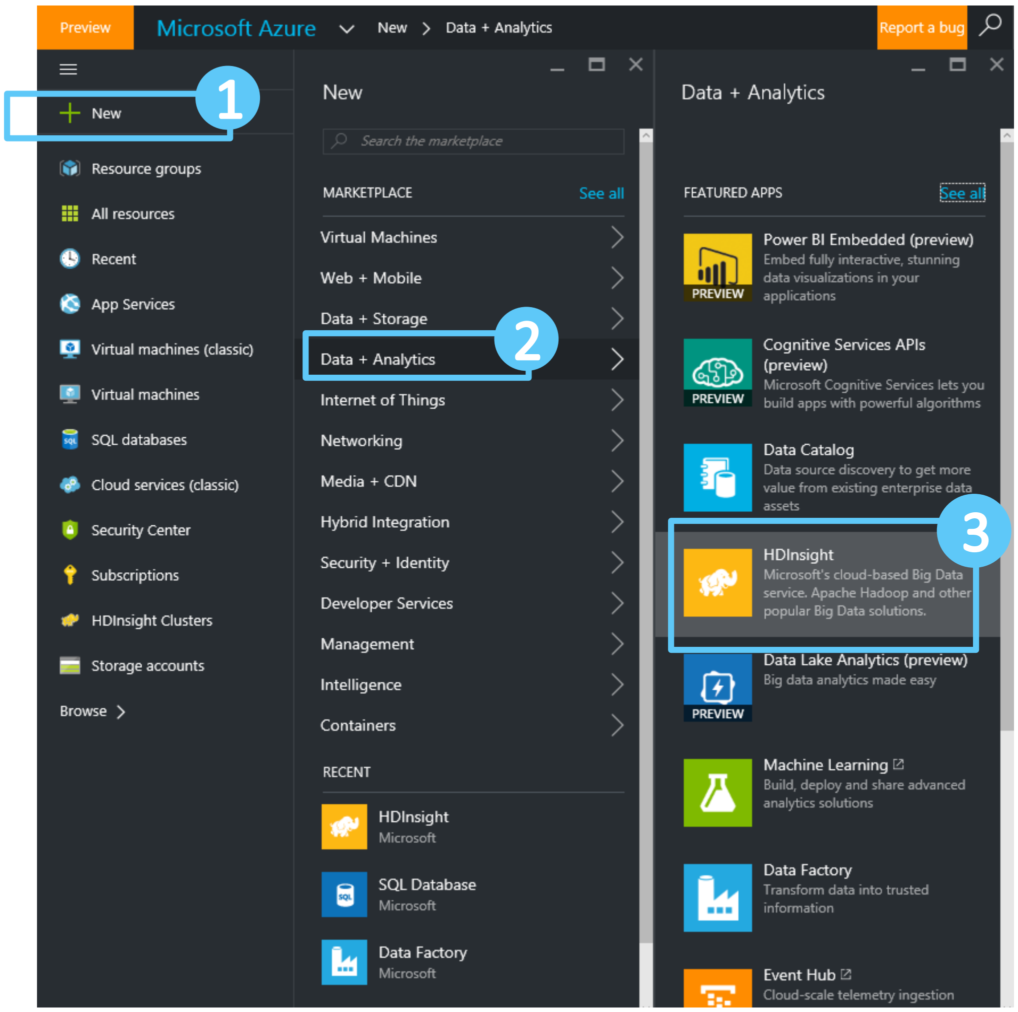
- **Credentials:** The HDInsight clusters allows to configure two user accounts during cluster creation:

- **HTTP user**: The default user name is admin using the basic configuration on the Azure Portal. Sometimes, it is called "Cluster user".
- **SSH User (Linux clusters):** Is used to connect to the cluster using SSH. We can use PuTTY in windows to use this cluster in a way similar to Terminal in Linux.

- **Data Source:** This option is to choose the Azure Storage Account and the Blob Storage Container to be used for HDInsight.
  - **Location**(Region): The HDInsight cluster and its default storage account must be located on the same Azure location.

- **Pricing:** Here, we can specify the no. of nodes in cluster and choose their configuration as well. The price of nodes varies based on its computation power.

- **Resource group name:** Azure Resource Manager (ARM) enables you to work with the resources in your application as a group, referred to as an Azure Resource Group. You can deploy, update, monitor or delete all of the resources for your application in a single, coordinated operation.



Embed fully interactive, stunning data visualizations in your applications

Microsoft Cognitive Services lets you build apps with powerful algorithms

Data source discovery to get more value from existing enterprise data assets

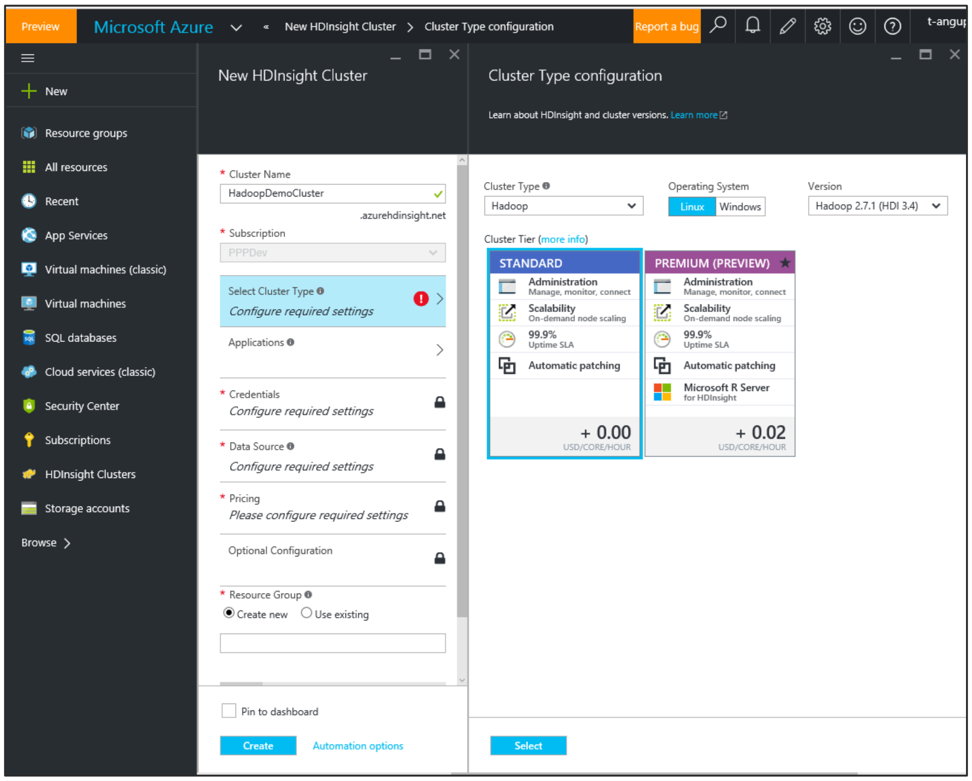
Microsoft's cloud-based Big Data service. Apache Hadoop and other popular Big Data solutions.

Big data analytics made easy

Build, deploy and share advanced analytics solutions

Transform data into trusted information

Cloud-scale telemetry ingestion



Preview | Microsoft Azure | « New HDInsight Cluster > Cluster Type configuration | Report a bug | t-angu

New | Resource groups | All resources | Recent | App Services | Virtual machines (classic) | Virtual machines | SQL databases | Cloud services (classic) | Security Center | Subscriptions | HDInsight Clusters | Storage accounts | Browse >

New HDInsight Cluster

* Cluster Name: HadoopDemoCluster ✓ .azurehdinsight.net

* Subscription: PPPDev

Select Cluster Type: *Configure required settings*

Applications

* Credentials: *Configure required settings*

* Data Source: *Configure required settings*

* Pricing: *Please configure required settings*

Optional Configuration

* Resource Group: (● Create new) ○ Use existing

☐ Pin to dashboard

Create | Automation options

Cluster Type configuration

Learn about HDInsight and cluster versions. Learn more

Cluster Type: Hadoop

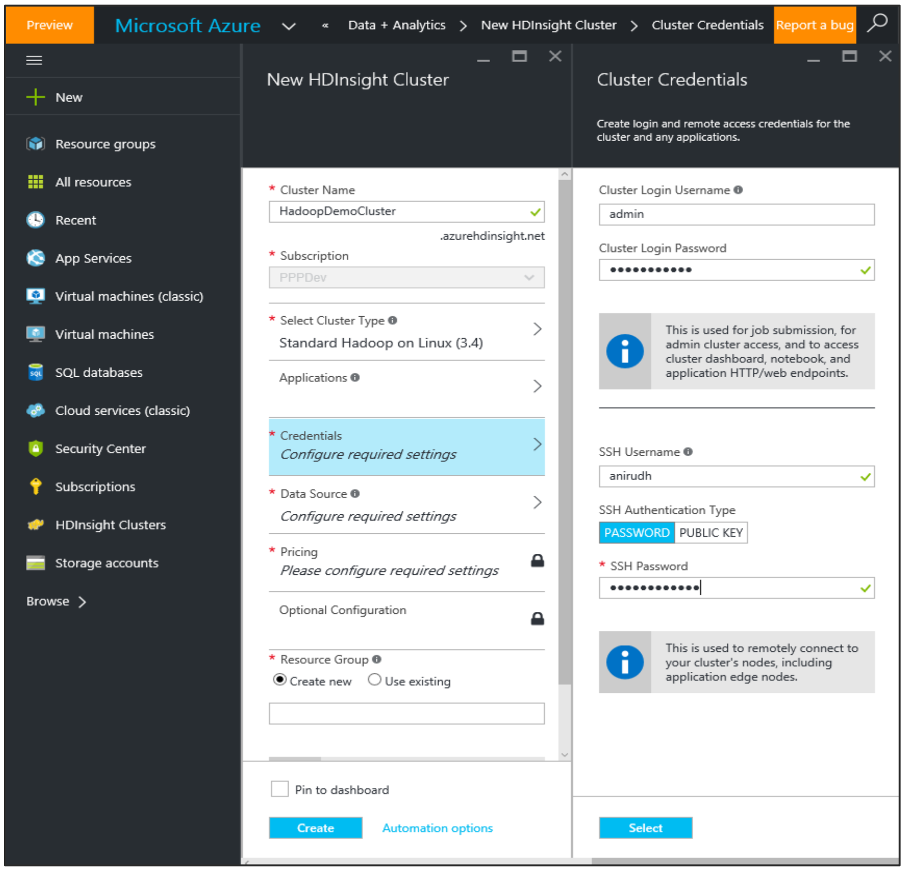
Operating System: [✓ Linux] Windows

Version: Hadoop 2.7.1 (HDI 3.4)

Cluster Tier (more info)

STANDARD	PREMIUM (PREVIEW)
Administration — Manage, monitor, connect	Administration — Manage, monitor, connect
Scalability — On-demand node scaling	Scalability — On-demand node scaling
99.9% Uptime SLA	99.9% Uptime SLA
Automatic patching	Automatic patching
	Microsoft R Server for HDInsight
+ 0.00 USD/CORE/HOUR	+ 0.02 USD/CORE/HOUR

Select



Preview | Microsoft Azure | « Data + Analytics > New HDInsight Cluster > Cluster Credentials | Report a bug

New | Resource groups | All resources | Recent | App Services | Virtual machines (classic) | Virtual machines | SQL databases | Cloud services (classic) | Security Center | Subscriptions | HDInsight Clusters | Storage accounts | Browse >

New HDInsight Cluster

* Cluster Name: HadoopDemoCluster ✓ .azurehdinsight.net

* Subscription: PPPDev

* Select Cluster Type: Standard Hadoop on Linux (3.4)

Applications

* Credentials: *Configure required settings*

* Data Source: *Configure required settings*

* Pricing: *Please configure required settings*

Optional Configuration

* Resource Group: (● Create new) ○ Use existing

☐ Pin to dashboard

Create | Automation options

Cluster Credentials

Create login and remote access credentials for the cluster and any applications.

Cluster Login Username: admin

Cluster Login Password: ●●●●●●●●●●● ✓

This is used for job submission, for admin cluster access, and to access cluster dashboard, notebook, and application HTTP/web endpoints.

SSH Username: anirudh ✓

SSH Authentication Type: [✓ PASSWORD] PUBLIC KEY

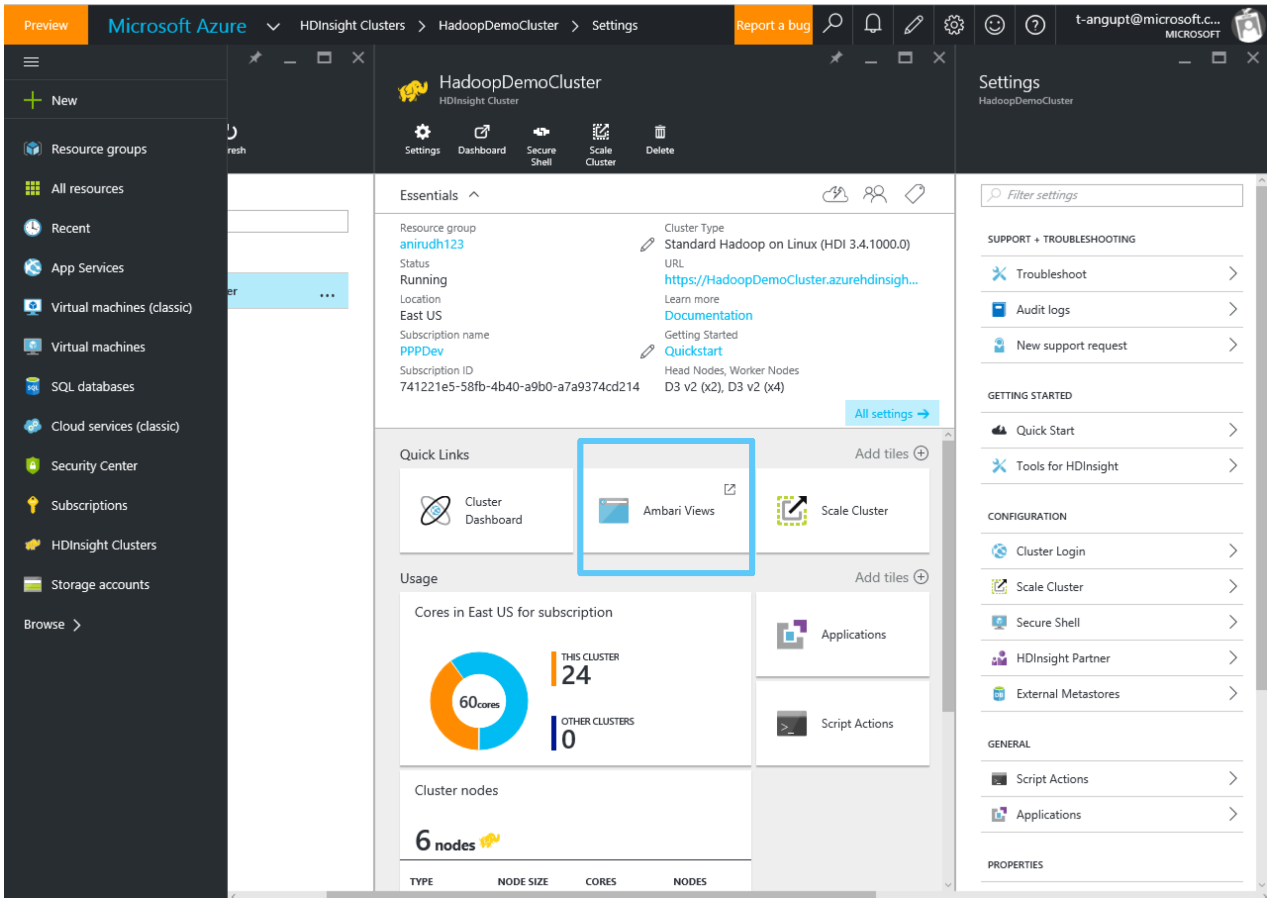
* SSH Password: ●●●●●●●●●●●● ✓

This is used to remotely connect to your cluster's nodes, including application edge nodes.

Select

Once you are done with configuring your cluster, click the create button to deploy the cluster. It will take around 20-25 minutes for your cluster to be deployed.

After deploying your cluster, move to the cluster in Azure. The screen will look like



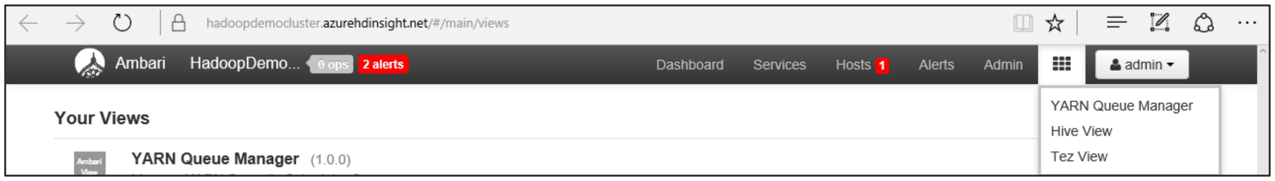
Resource group: anirudh123
Status: Running
Location: East US
Subscription name: PPPDev
Subscription ID: 741221e5-58fb-4b40-a9b0-a7a9374cd214
Cluster Type: Standard Hadoop on Linux (HDI 3.4.1000.0)
URL: https://HadoopDemoCluster.azurehdinsigh...
Learn more: Documentation
Getting Started: Quickstart
Head Nodes, Worker Nodes: D3 v2 (x2), D3 v2 (x4)

Cores in East US for subscription: 60 cores; THIS CLUSTER 24; OTHER CLUSTERS 0

Cluster nodes: 6 nodes

TYPE	NODE SIZE	CORES	NODES

The cluster can be scaled based on the need after the creation by using the Scale Cluster Tab on the HDInsight window. For working on Hadoop one can either use Ambari or ssh to the cluster on PuTTy. Ambari is an open-source tool present in HDInsight to manage and monitor Hadoop using a GUI interface. On HDInsight, the Ambari can be used by clicking on the Ambari Tab and providing the HTTP (**admin**) credentials.



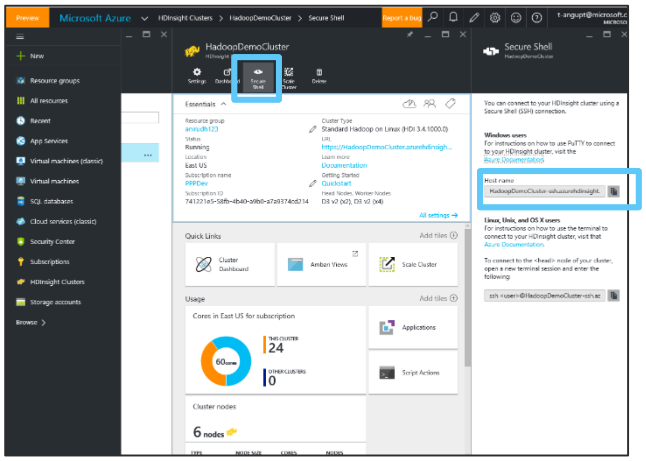
Your Views

YARN Queue Manager (1.0.0)

USING PUTTY

We can use Secure Shell (SSH) to connect to the HDInsight Cluster. For this we need a SSH Client. Linux, Unix, and Mac OS come with an SSH client. Windows users can download a client, such as PuTTY. Enter the SSH Hostname on PuTTY and login with the ssh credentials of HDInsight Credentails (Login only when the deployment is successful).

PuTTY Download Link:



Resource group: anirudh123
Status: Running
Location: East US
Subscription name: PPPDev
Subscription ID: 741221e5-58fb-4b40-a9b0-a7a9374cd214
Cluster Type: Standard Hadoop on Linux (HDI 3.4.1000.0)
URL: https://HadoopDemoCluster.azurehdinsigh...
Learn more: Documentation
Getting Started: Quickstart
Head Nodes, Worker Nodes: D3 v2 (x2), D3 v2 (x4)

Cores in East US for subscription: 60 cores; THIS CLUSTER 24; OTHER CLUSTERS 0
Cluster nodes: 6 nodes

You can connect to your HDInsight cluster using a Secure Shell (SSH) connection.

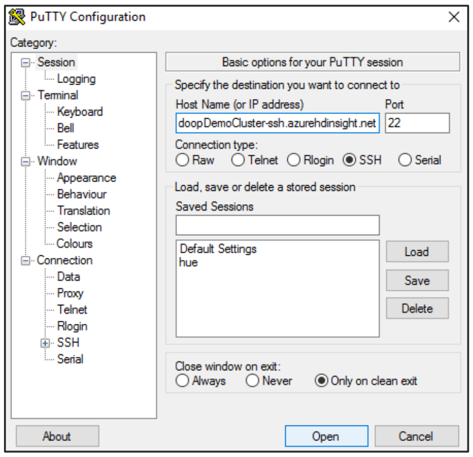
Windows users
For instructions on how to use PuTTY to connect to your HDInsight cluster, visit the Azure Documentation.

Host name: HadoopDemoCluster-ssh.azurehdinsight.

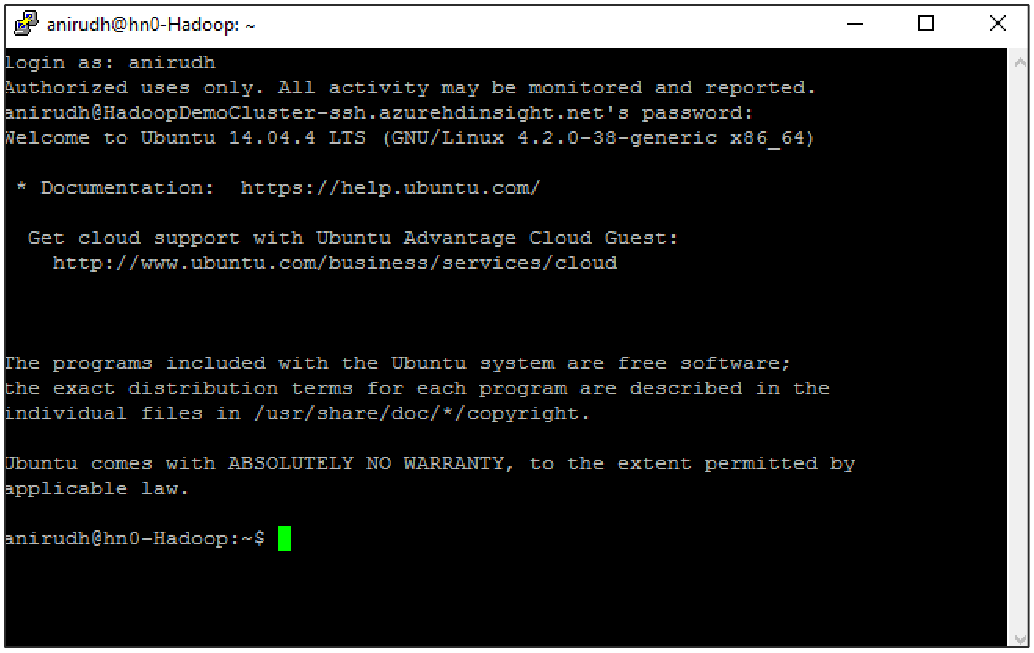
Linux, Unix, and OS X users
For instructions on how to use the terminal to connect to your HDInsight cluster, visit that Azure Documentation.

To connect to the <head> node of your cluster, open a new terminal session and enter the following:

ssh <user>@HadoopDemoCluster-ssh.az



Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): doopDemoCluster-ssh.azurehdinsight.net
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: Default Settings, hue
Close window on exit: Always Never [✓ Only on clean exit]



```
anirudh@hn0-Hadoop: ~
login as: anirudh
Authorized uses only. All activity may be monitored and reported.
anirudh@HadoopDemoCluster-ssh.azurehdinsight.net's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-38-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

anirudh@hn0-Hadoop:~$
```

PuTTY Download Link: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

FEW HADOOP SHELL COMMANDS

Listed below are few shell commands which can be used using PuTTY. As the default storage for HDInsight is Blob storage, some of the commands help in using the blob storage from the terminal. Here **wasb://<container>@<storage_account>.blob.core.net/folder1/abc.txt** is used to refer to the file name abc.txt in folder1 directory on the blob store. It can also be referred by **wasb:///folder1/abc.txt** (if the container is the primary storage of the cluster).

Command	Description
`> nano file.txt`	Opens the nano editor to create/edit a file.
`> hadoop dfs -ls wasb:///folder1`	Lists out all the directories and files.
`> hadoop dfs -cat wasb:///file1.txt`	Prints the content of the file on the terminal.
`> hadoop dfs -mkdir wasb:///home/dir1`	Creates a directory. Here new directory **dir1** would be created in the home directory of the default storage blob.
`> hadoop dfs -mkdir -p wasb:///dir1/dir2/dir3`	-p is used to create to create the parent directories also if they don't exist.
`> hadoop dfs -put localfile1 localfile2 wasb:///user/hadoop`	Copies the file or files to the specified blob store location. Not restricted to local files only.
`> hadoop dfs -copyFromLocal local1.txt wasb:///dir1/local1.txt`	Copies the local file to the specified location. Restricted to local file only.
`> hadoop dfs -copyToLocal wasb:///dir1/local1.txt  file1.txt`	Copies the file on hdfs to local file system.
`> hadoop dfs -moveFromLocal localfile wasb:///dir1/`	Moves the local file to the specified location.
`> hadoop dfs -rmr wasb:///file1.txt`	Deletes the files and sub directories.

Here, `localfile` file refers to the file on the cluster node.
You can use winscp to move the file from local machine to the cluster node.
Winscp download lnk: https://winscp.net/eng/download.php

APACHE HIVE

A SQL-like Data processing tool.

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Since Hadoop is **"Schema on Read"** Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive. Since HDInsight is a PaaS service, this tool is readily available.

Hive, initially was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It allows to project structure on largely unstructured data. After the structure is defined, we can use Hive to query that data without knowledge of Java or MapReduce. It is basically a high-level tool for MapReduce. **HiveQL** (the Hive query language) allows you to write queries with statements that are similar to T-SQL (SQL like).

Hive is NOT-

- A relational database.
- A design for On-Line Transaction Processing (OLTP).
- A language for real-time data streaming queries.

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It is a High-level tool for MapReduce.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.
- UDF's can in other languages.

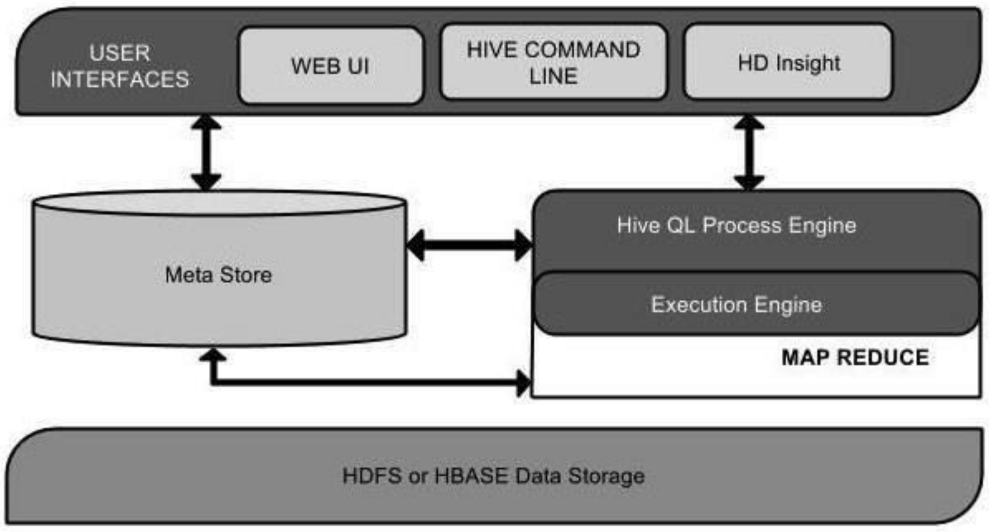


Fig: Hive Architecture

HIVE INTERNAL & EXTERNAL TABLES

There are a few things one should know about the Hive internal table and external table:

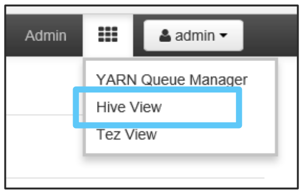
- The **CREATE TABLE** command creates an internal table. The data file must be located in the default container.
- The **CREATE TABLE** command moves the data file to the */hive/warehouse/ folder.*
- The **CREATE EXTERNAL TABLE** command creates an external table. The data file can be located outside the default container.
- The **CREATE EXTERNAL TABLE** command does not move the data file.
- The **CREATE EXTERNAL TABLE** command doesn't allow any folders in the LOCATION.

Link: https://blogs.msdn.microsoft.com/cindygross/2013/02/05/hdinsight-hive-internal-and-external-tables-intro/

PROCESSING DATA USING HIVE (AMBARI)

Ambari is a management and monitoring utility provided with Linux-based HDInsight clusters. One of the features provided through Ambari is a Web UI that can be used to run Hive queries.

To use HIVE, move to the Ambari View then select the set of squares from the page menu (next to the **Admin** link and button on the left of the page,) to list available views. Select the **Hive view**. You will see a similar screen.



Hive Commands: http://www.edureka.co/blog/hive-commands-with-examples

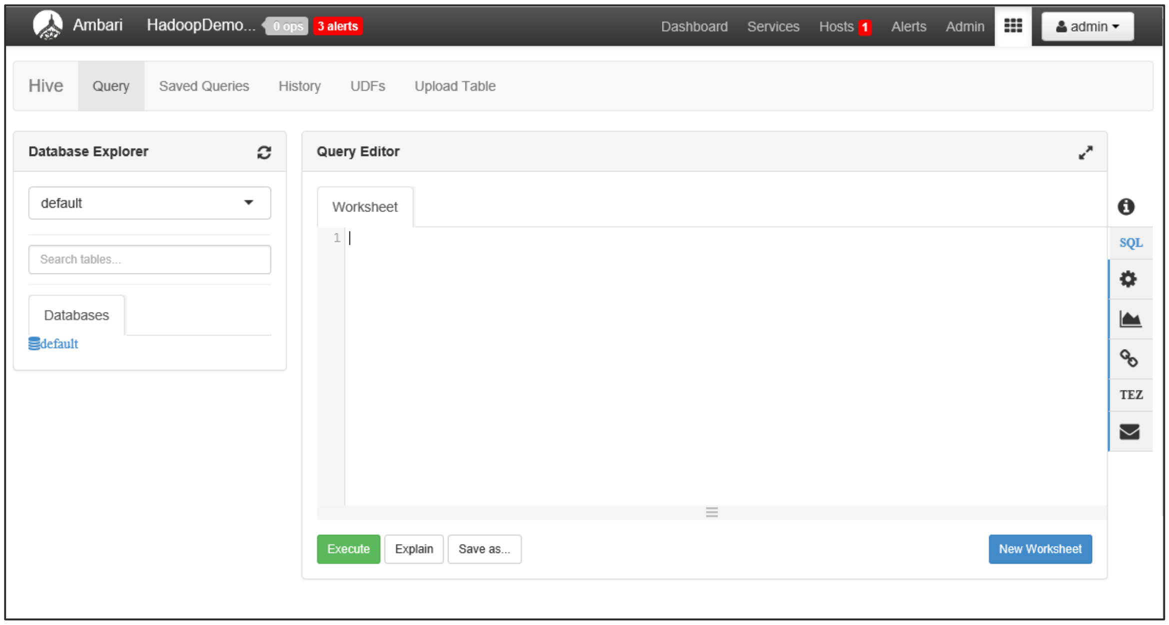
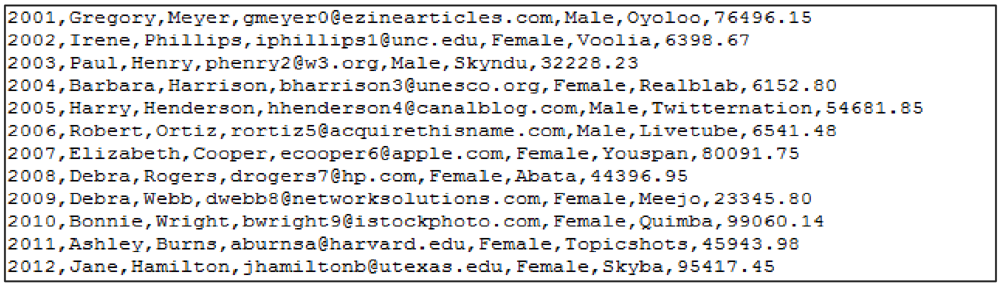


Fig: Hive View in Ambari

- **CREATING A SIMPLE HIVE TABLE-**

1. Create a Dataset and Store it in Blob Store. (Below is sample data **MOCK_DATA.csv** with fields separated by **comma (,).**



```
2001,Gregory,Meyer,gmeyer0@ezinearticles.com,Male,Oyoloo,76496.15
2002,Irene,Phillips,iphillips1@unc.edu,Female,Voolia,6398.67
2003,Paul,Henry,phenry2@w3.org,Male,Skyndu,32228.23
2004,Barbara,Harrison,bharrison3@unesco.org,Female,Realblab,6152.80
2005,Harry,Henderson,hhenderson4@canalblog.com,Male,Twitternation,54681.85
2006,Robert,Ortiz,rortiz5@acquirethisname.com,Male,Livetube,6541.48
2007,Elizabeth,Cooper,ecooper6@apple.com,Female,Youspan,80091.75
2008,Debra,Rogers,drogers7@hp.com,Female,Abata,44396.95
2009,Debra,Webb,dwebb8@networksolutions.com,Female,Meejo,23345.80
2010,Bonnie,Wright,bwright9@istockphoto.com,Female,Quimba,99060.14
2011,Ashley,Burns,aburnsa@harvard.edu,Female,Topicshots,45943.98
2012,Jane,Hamilton,jhamiltonb@utexas.edu,Female,Skyba,95417.45
```

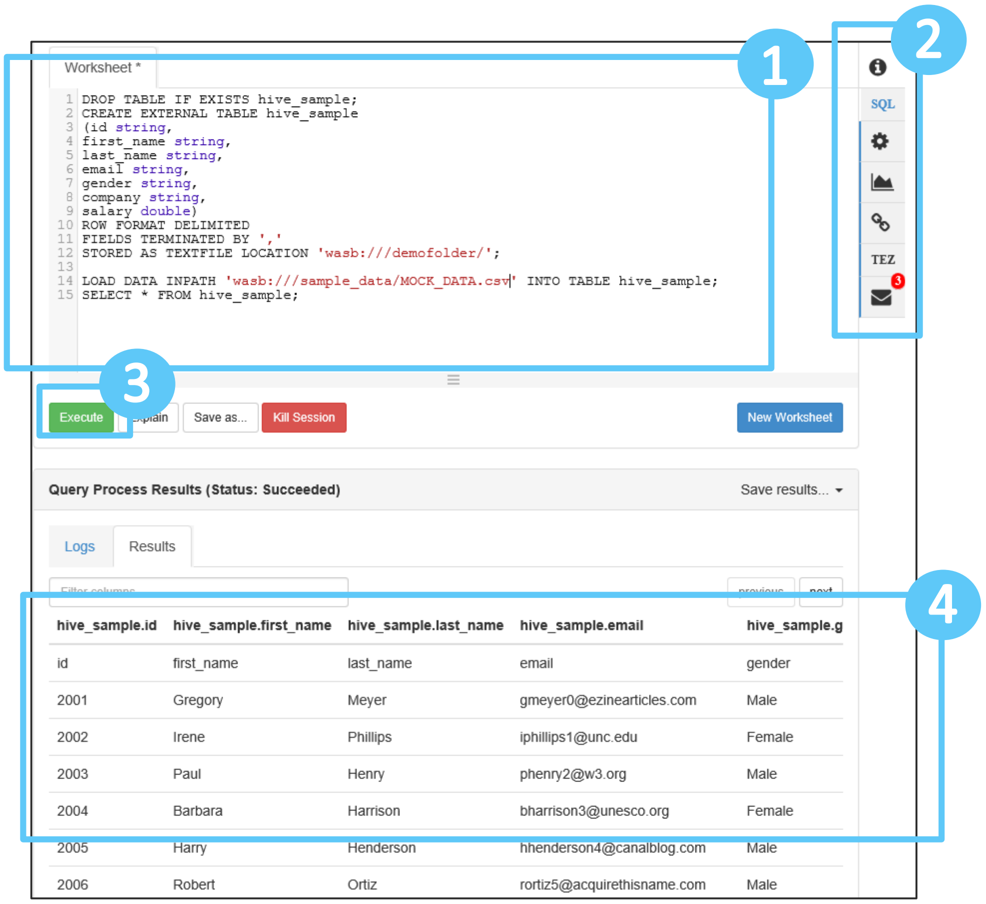
2. Write the following Hive Query, in the query Editor.

```
DROP TABLE IF EXISTS hive_sample;
CREATE EXTERNAL TABLE hive_sample
(id string,
first_name string,
last_name string,
email string,
gender string,
company string,
salary double)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 'wasb:///demofolder';

LOAD DATA INPATH 'wasb:///sample_data/MOCK_DATA.csv' INTO TABLE
hive_sample;
```

- **DROP TABLE** - Deletes the table and the data file, in case the table already exists.
- **CREATE EXTERNAL TABLE** - Creates a new "external" table in Hive. External tables store only the table definition in Hive; the data is left in the original location.
- **ROW FORMAT** - Tells Hive how the data is formatted. In this case, the fields in each field are separated by `','`.
- **STORED AS TEXTFILE LOCATION** - Tells Hive where the data is stored (the example/data directory), and that it is stored as text.
- **LOAD DATA**- Loads data in the given path to the specified table (if we want all the files in the given folder to be loaded just mention the folder path, `'wasb:///sample_data/')`

This will create a table a table and load data from the file to the table. Now the data can be processed with hive queries written in HQL (like SQL). We can save the queries for later use.

Link: https://azure.microsoft.com/en-in/documentation/articles/hdinsight-hadoop-use-hive-ambari-view/



```
DROP TABLE IF EXISTS hive_sample;
CREATE EXTERNAL TABLE hive_sample
(id string,
first_name string,
last_name string,
email string,
gender string,
company string,
salary double)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 'wasb:///demofolder/';

LOAD DATA INPATH 'wasb:///sample_data/MOCK_DATA.csv' INTO TABLE hive_sample;
SELECT * FROM hive_sample;
```

hive_sample.id	hive_sample.first_name	hive_sample.last_name	hive_sample.email	hive_sample.g
id	first_name	last_name	email	gender
2001	Gregory	Meyer	gmeyer0@ezinearticles.com	Male
2002	Irene	Phillips	iphillips1@unc.edu	Female
2003	Paul	Henry	phenry2@w3.org	Male
2004	Barbara	Harrison	bharrison3@unesco.org	Female
2005	Harry	Henderson	hhenderson4@canalblog.com	Male
2006	Robert	Ortiz	rortiz5@acquirethisname.com	Male

1. HQL Query Editor
2. Tool Bar
   - SQL for Hive Editor
   - Hive Settings
   - Visual Explain
   - TEZ- to view DAGs
   - Notifications
3. Execute Button
4. The Result Panel

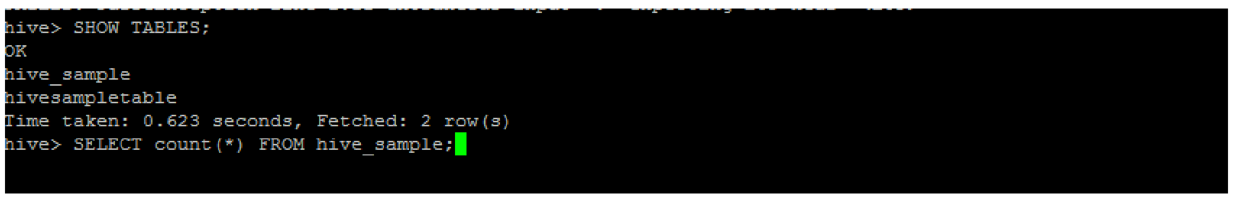
Fig: Hive View in Ambari

PROCESSING DATA USING HIVE (PuTTY)

Hive can also be accessed using ssh client (PuTTY). There are 2 ways to do so:

- **HIVE CLI**- A SQL like command line interface for executing hive instructions one by one.
- **HADOOP Command Line-** This is used to run the Hive query file (.hql) from the Hadoop Command Line.

- ### HIVE COMMAND LINE INTERFACE (CLI)-
  1. Use PuTTY to connect with your HDInsight Cluster
  2. Type **`hive`** and press Enter to open the CLI
  3. Type the statements you need to execute here.



```
hive> SHOW TABLES;
OK
hive_sample
hivesampletable
Time taken: 0.623 seconds, Fetched: 2 row(s)
hive> SELECT count(*) FROM hive_sample;
```

- **HADOOP COMMAND LINE -**

1. Use PuTTY to connect with your HDInsight Cluster
2. Create a new query file named `hive_query1.hql` This will open the nano editor to create the file.

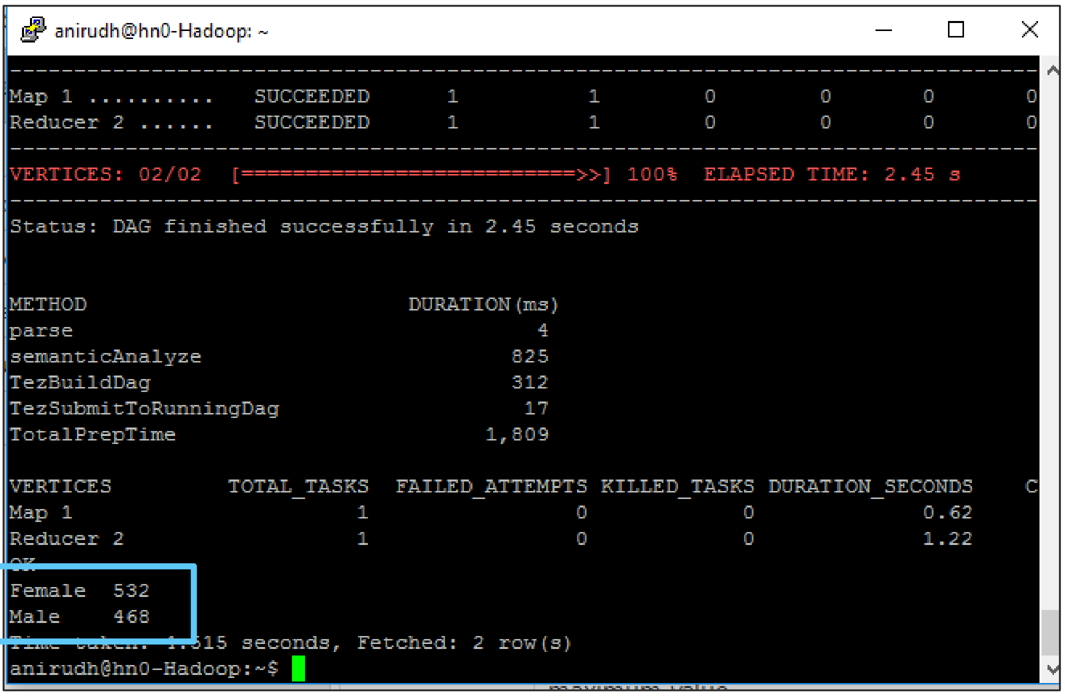
```
nano hive_query1.hql
```

3. Enter the following statements to create a new table and insert data to it from the existing `hive_sample` table. The result would be the count of Male and Female clients.

```
DROP TABLE IF EXISTS test_table_1;
CREATE TABLE test_table_1
(client_id string,
name string,
gender string,
salary double);

INSERT INTO test_table_1
SELECT id, first_name, gender,salary FROM hive_sample;

SELECT gender, count(*) FROM test_table_1 GROUP BY gender;
```

4. Press `Ctrl + X` to save and Exit from the Nano Editor.
5. To execute the query file, type the following

```
hive -f hive_query1.hql
```



```
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0        0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0        0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 2.45 s
--------------------------------------------------------------------------------
Status: DAG finished successfully in 2.45 seconds


METHOD                         DURATION(ms)
parse                                    4
semanticAnalyze                        825
TezBuildDag                            312
TezSubmitToRunningDag                   17
TotalPrepTime                        1,809

VERTICES         TOTAL_TASKS  FAILED_ATTEMPTS KILLED_TASKS DURATION_SECONDS    C
Map 1                      1                0            0             0.62
Reducer 2                  1                0            0             1.22
OK
Female  532
Male    468
Time taken: 1.515 seconds, Fetched: 2 row(s)
anirudh@hn0-Hadoop:~$
```

The result of Hive query can also be stored in an external file on Blob Store using,

```
INSERT OVERWRITE DIRECTORY 'wasb:///data/result1.csv' SELECT * FROM test_table_1;
```

Suppose you have created an internal table on hive and stored data in it. The data file will be moved to **hive/warehouse/table_name** directory. Now let's say you were done with your work and you deleted the cluster. The next time if you create a cluster on same blob store and create a hive table with same name as before you will get all the data that you stored previously (if the schema differs then the table will conatin NULL's and some of the previous data).

For more information, visit**:** https://cwiki.apache.org/confluence/display/Hive/LanguageManual

Some Facts

- **HiveQL** is SQL-like, not exactly SQL. So some commands are not supported like DELETE.
- **HiveQL** supports indexing and partitioning.
- Hive can also store data in an efficient Row Columnar way called **ORC** (Optimizied Row Columnar).
  Visit: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

APACHE PIG

High Level MapReduce for ETL jobs

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig. In **2006**, Apache Pig was developed as a research project at Yahoo, especially to create and execute MapReduce jobs on every dataset. In **2007**, Apache Pig was open sourced via Apache incubator. In **2008**, the first release of Apache Pig came out. In **2010**, Apache Pig graduated as an Apache top-level project.

To write data analysis programs, HDInsight comes with pre-loaded Pig tool. Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Features of PIG

- Rich set of operators with parallel operations.
- A procedural language.
- Pig Latin is similar to SQL.
- Automatic Optimization of Tasks.
- **User-defined Functions(UDF)** can be created in other languages and invoked.
- Handles all kinds of data, both **structured as well as unstructured**.
- It is a procedural language and fits in pipeline paradigm.
- Provides operators to perform ETL (Extract, Transform, and Load) functions.

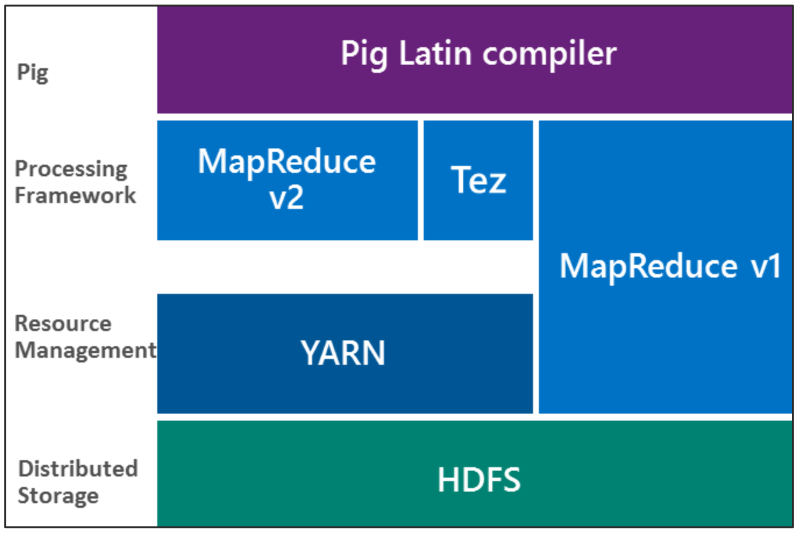
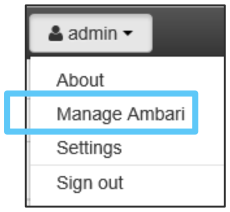


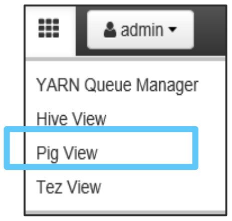
Fig: Pig Architecture

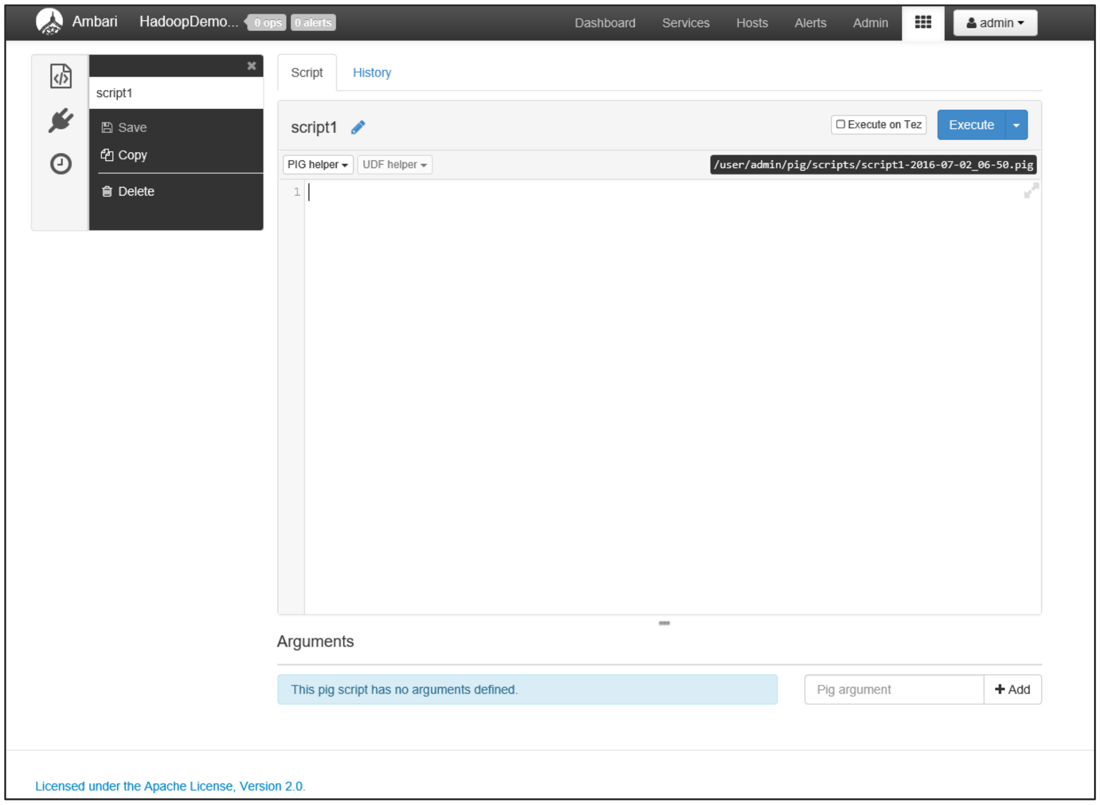
PROCESSING DATA USING PIG (AMBARI)

Since PIG View is not available in Ambari by default, we need to create an instance of Pig from the Manage Ambari section. Following steps will help in doing this.



1. Click on the **admin** tab in the navigation toolbar of Ambari.
2. Select the **Manage Ambari** option from the list, a window will appear.
3. Click on the **View** Button present in the Deploy View section.
4. Click on the **Pig** tab and select ➕ Create Instance
5. Fill in the Details and click on the **save** button
   - Instance Name: Pig
   - Display Name: Pig View
   - Description: My new Pig View
6. Notice that the **Pig View** option appears just below the Hive View.
7. Click on **New Script** button to create a new pig script.





/user/admin/pig/scripts/script1-2016-07-02_06-50.pig

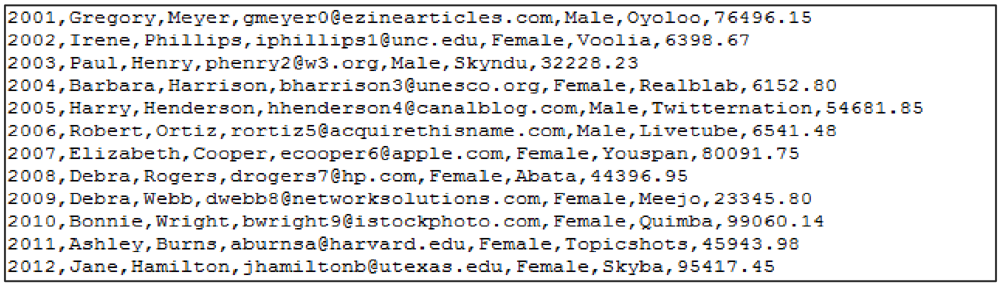
This pig script has no arguments defined.

Fig: Pig View in Ambari

The scripts created in Ambari are saved at ***<blob-container>/user/admin/pig/scripts*** location

- **CREATING A SIMPLE PIG JOB-**

1. We would be using the same dataset used for performing Hive operations (MOCK_DATA.csv).

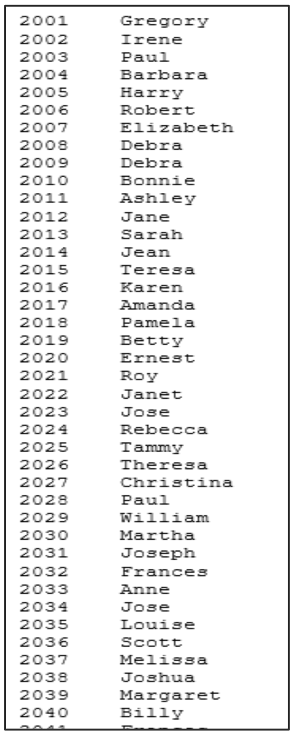


```
2001,Gregory,Meyer,gmeyer0@ezinearticles.com,Male,Oyoloo,76496.15
2002,Irene,Phillips,iphillips1@unc.edu,Female,Voolia,6398.67
2003,Paul,Henry,phenry2@w3.org,Male,Skyndu,32228.23
2004,Barbara,Harrison,bharrison3@unesco.org,Female,Realblab,6152.80
2005,Harry,Henderson,hhenderson4@canalblog.com,Male,Twitternation,54681.85
2006,Robert,Ortiz,rortiz5@acquirethisname.com,Male,Livetube,6541.48
2007,Elizabeth,Cooper,ecooper6@apple.com,Female,Youspan,80091.75
2008,Debra,Rogers,drogers7@hp.com,Female,Abata,44396.95
2009,Debra,Webb,dwebb8@networksolutions.com,Female,Meejo,23345.80
2010,Bonnie,Wright,bwright9@istockphoto.com,Female,Quimba,99060.14
2011,Ashley,Burns,aburnsa@harvard.edu,Female,Topicshots,45943.98
2012,Jane,Hamilton,jhamiltonb@utexas.edu,Female,Skyba,95417.45
```

2. Type the following statements in the script editor pane.

```
data = LOAD 'wasb:///sample_data/MOCK_DATA.csv' USING PigStorage(',');
filter_data = FILTER data BY $0 < 2050;
select_data = FOREACH data GENERATE $0, $1;
STORE select_data INTO 'wasb:///pig_result/1/' USING PigStorage('\t');
DUMP filter data;
```

- **LOAD** - It loads the data from the file location into the variable.
- **USING PIGSTORAGE(',')**- Tells how the columns are separated, comma(',') separated in our case
- **FILTER** - Similar to where clause in SQL. Used to select the required tuples from a relation based on a condition. Here $0 is used to denote the first column.
- **FOREACH** -It is used to generate specified data transformations based on the column data.
- **STORE**- It stores the variable data to the specified location separated by the given operator (tab space in our case).
- **DUMP**- It used to display the variable data on the standard output (std out).



```
2001     Gregory
2002     Irene
2003     Paul
2004     Barbara
2005     Harry
2006     Robert
2007     Elizabeth
2008     Debra
2009     Debra
2010     Bonnie
2011     Ashley
2012     Jane
2013     Sarah
2014     Jean
2015     Teresa
2016     Karen
2017     Amanda
2018     Pamela
2019     Betty
2020     Ernest
2021     Roy
2022     Janet
2023     Jose
2024     Rebecca
2025     Tammy
2026     Theresa
2027     Christina
2028     Paul
2029     William
2030     Martha
2031     Joseph
2032     Frances
2033     Anne
2034     Jose
2035     Louise
2036     Scott
2037     Melissa
2038     Joshua
2039     Margaret
2040     Billy
```

The figure is a snapshot of the file at `'wasb:///pig_result/1/'` which is the data of `select_data` variable. It is similar to the Select query in SQL.

We can also specify the schema to the data using **AS** operator. In above script we have not given any schema to our data, which is a way to handle semi-structured or un-structured data using Pig.

For more Built-in Pig Functions visit-
https://pig.apache.org/docs/r0.9.1/func.html#count
http://www.tutorialspoint.com/apache_pig/apache_pig_reading_data.html

PROCESSING DATA USING PIG (PuTTY)

Similar to HIVE, pig also has its own command line interface (CLI).

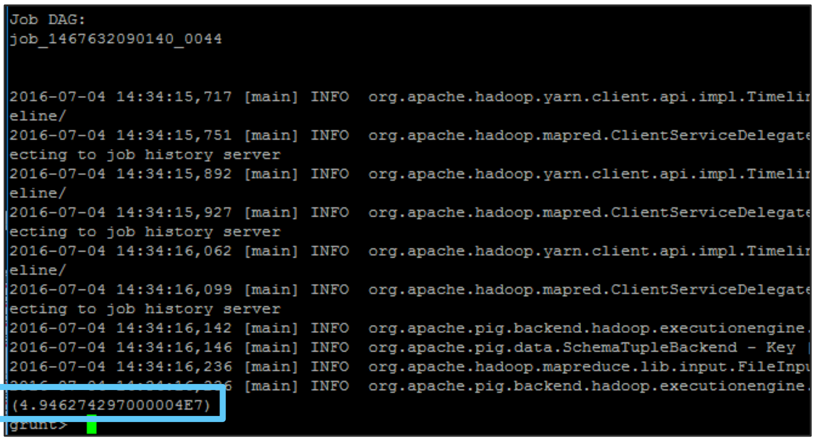
- ### PIG GRUNT SHELL (CLI)-

  The PIG grunt shell executes command one-by-one. The following is a simple script to calculate the total of the salary column. Notice the schema defined for the Data.

  1. Use PuTTY to connect with your HDInsight Cluster
  2. Type `pig` and press Enter to open the CLI
  3. Type the following statements

```
data = LOAD 'wasb:///sample_data/MOCK_DATA.csv' USING PigStorage(',') As
(id:chararray,
first_name:chararray,
last_name:chararray,
email:chararray,
gender:chararray,
company:chararray,
salary:double);

sal = FOREACH data GENERATE salary;
grouped_data = GROUP sal ALL;
total_sal = FOREACH grouped_data GENERATE SUM(sal);

DUMP total_sal;
```

- **LOAD -** It loads the data in the defined schema. The first column will be refered as id, second as first_name.
- **GROUP ALL -** The **group** operator is used to group the data in one or more relations. It collects the data having the same key. The **group all** operator is used to group all the records of the relation.
- **SUM ()-** adds the values belonging to the same group.



```
Job DAG:
job_1467632090140_0044


2016-07-04 14:34:15,717 [main] INFO  org.apache.hadoop.yarn.client.api.impl.Timelir
eline/
2016-07-04 14:34:15,751 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate
ecting to job history server
2016-07-04 14:34:15,892 [main] INFO  org.apache.hadoop.yarn.client.api.impl.Timelir
eline/
2016-07-04 14:34:15,927 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate
ecting to job history server
2016-07-04 14:34:16,062 [main] INFO  org.apache.hadoop.yarn.client.api.impl.Timelir
eline/
2016-07-04 14:34:16,099 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate
ecting to job history server
2016-07-04 14:34:16,142 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
2016-07-04 14:34:16,146 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key
2016-07-04 14:34:16,236 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInpu
2016-07-04 14:34:16,2[illegible]6 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
(4.946274297000004E7)
grunt>
```

Link: https://www.youtube.com/watch?v=FvRNfLnxHmE

- **PIG COMMAND LINE**

Similar to HIVE we can create a pig script file using any text editor and run it using the hadoop command line. The following command in used to run the script (`script_1.pig`).

```
pig -x tez script_1.pig
```

- **Note** – hdfs commands like (hadoop -mkdir, hadoop -rmr) can also be included in the pig script. Pig fails to store output in the existing location and gives an error. These commands can be used to solve this problem.

For more information, visit:

- http://hortonworks.com/hadoop-tutorial/how-to-use-basic-pig-commands/
- http://pig.apache.org/docs/r0.16.0/start.html

Some Facts

- Hive was designed to appeal to a community comfortable with SQL. Its philosophy was that we don't need yet another scripting language. Hive supports map and reduce transform scripts in the language of the user's choice (which can be embedded within SQL clauses).
- The main motive behind developing Pig was to cut-down on the time required for development via its multi query approach. Pig is a high level data flow system that renders you a simple language platform popularly known as Pig Latin that can be used for manipulating data and queries.

HIVE VS PIG

HIVE	PIG
Uses HiveQL	Uses PigLatin
Declarative Language	Procedural Language
A Data warehouse tool	A Data Flow Language
For Structured Data	For Structured and Semi-Structured Data
Needs Meta Data (schema etc.)	No Meta Data Required
Supports partitions and indexes	No such support
Supports JOINS, ORDER, SORT	Supports JOINS, ORDER, SORT
Supports External UDF’s	Supports External UDF’s
Supports JDBC/ODBC(limited)	No JDBC/ODBC support
Use if there are Limited number of JOINS and Filters	Recommendable when there are huge number of JOINs and Filters
Mainly used for creating Reports	Mainly used for Programming
Operates on Server Side of any cluster	Operates on Client Side
Uses: • Ad-Hoc queries • Business Intelligence Analysis	Uses: • Data Pipelines • Iterative Processing • Research

Links: https://developer.yahoo.com/blogs/hadoop/pig-hive-yahoo-464.html
https://www.quora.com/What-is-the-difference-between-Hadoop-HIVE-and-PIG
For UDF’s visit- https://azure.microsoft.com/en-us/documentation/articles/hdinsight-python/

APACHE SQOOP

Eases movement of data IN and OUT of Hadoop

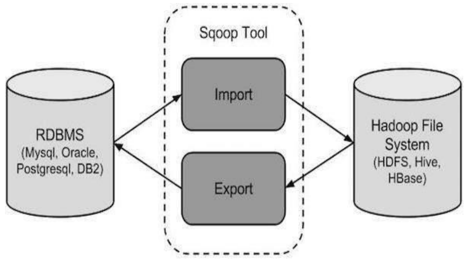
SQOOP is an open source which is the product of Apache. **SQOOP** stands for **SQ**L to Had**oop**. It is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as SQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

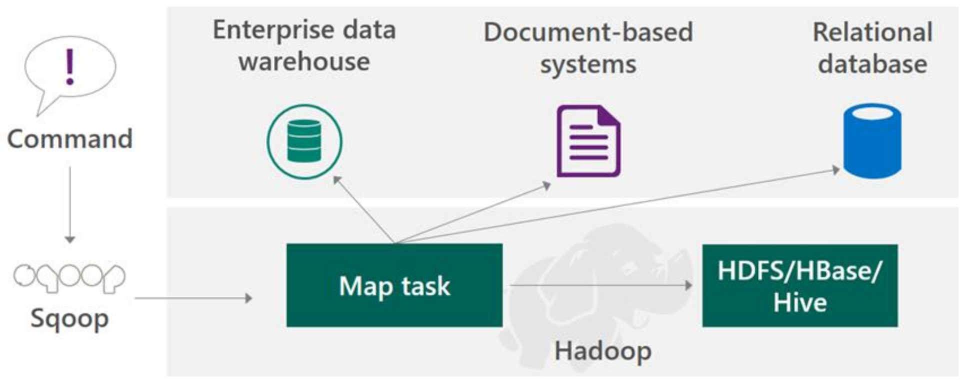
It uses Map reduce to fetch data from RDBMS and stores that on HDFS. By default, it uses four mappers but this value is configurable. It's recommended not to set higher number of mappers, as it may lead to consuming all spool space of the source database.

Sqoop ships as one binary package however it's compound from two separate parts - client and server. You need to install server on single node in your cluster. This node will then serve as an entry point for all connecting Sqoop clients. Server acts as a mapreduce client and therefore Hadoop must be installed and configured on machine hosting Sqoop server. Clients can be installed on any arbitrary number of machines. Client is not acting as a mapreduce client and thus you do not need to install Hadoop on nodes that will act only as a Sqoop client. Since, Sqoop comes as a part of Azure HDInsight, so there is no need for any installation.

Sqoop internally uses JDBC interface. There are two possible actions with sqoop:

- **Sqoop Import:** The import tool imports tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files.
- **Sqoop Export:** The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.





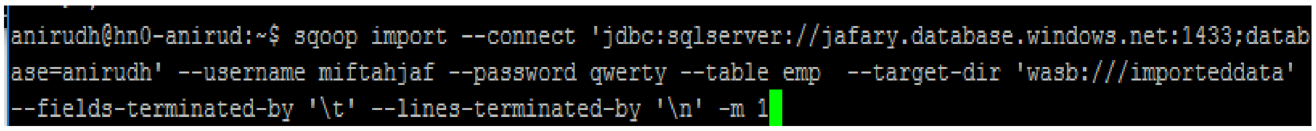
SQOOP IMPORT

- **IMPORT TABLE FROM SQL SERVER AS A FILE—**

  The syntax for transferring data from SQL server as a text file is –

```
sqoop import --connect '{SQL server address}' --username {username} --password
{pswd} --table {tbl_name} --target-dir {path} --fields-terminated-by '\t' --
lines-terminated-by '\n' -m 1
```

- ***{SQL server address}*** - the address the SQL server database.
  Eg- jdbc:sqlserver://jafary.database.windows.net: 1433; database=Anirudh
- ***{username}*** - the username used to access the database.
- ***{pswd}*** - the password for the entered username.
- ***{tbl_name}*** - the name of the table to be imported.
- ***{path}*** - the path where the imported table should be stored.
- The fields will be separated by '\t' and the records by '\n'.
- **-m 1** – specifies the number of mapper functions to be used to import data in parallel.



```
anirudh@hn0-anirud:~$ sqoop import --connect 'jdbc:sqlserver://jafary.database.windows.net:1433;datab
ase=anirudh' --username miftahjaf --password qwerty --table emp  --target-dir 'wasb:///importeddata'
--fields-terminated-by '\t' --lines-terminated-by '\n' -m 1
```

- **IMPORT TABLE FROM SQL SERVER AS A HIVE TABLE—**

The import tool can also be used to import the data from RDBMS to a HIVE table. The table is imported along with its schema and it becomes easier to apply various queries on the data.

Syntax-

```
sqoop import --connect '{SQL server address}' --username {username} --password
{pswd} --table {tbl_name} --hive-import -m 1
```

- --hive-import: this tells sqoop to import data along with its schema and store it as hive table.

SQOOP EXPORT

- ### EXPORT DATA FROM HDFS TO SQL SERVER—

Similar to the import, the data on HDFS can be exported to SQL server. The “export” command is used to achieve this.

Note: The schema of the table must exist on the SQL server

Syntax-

```
sqoop export --connect '{SQL server address}' --username {username} --password
{pswd} --table {tbl_name} --export-dir {path} -m 1
```

   - {*tbl_name*} - the name of the table in SQL server where the HDFS data should be exported
   - {*path*} - the path of the data to be exported

OTHER SQOOP ARGUMENTS

- ### LIST DATABASES IN SQL SERVER

```
sqoop list-databases --connect 'jdbc:sqlserver://miftah.database.windows.net:1433'
--username user1 --password Pwd@123
```

- ### LIST TABLES IN THE DATABASE ON SQL SERVER

```
sqoop list-tables --connect 'jdbc:sqlserver://miftah.database.windows.net:1433,
database=anirudh' --username user1 --password Pwd@123
```

- ### IMPORT ALL TABLES FROM DATABASE ON SQL SERVER

```
sqoop import-all-tables --connect
'jdbc:sqlserver://miftah.database.windows.net:1433, database=anirudh' --username
user1 --password Pwd@123 --exclude-tables table_2 --hive-import -m 1
```

- ### IMPORT SELECTED DATA FROM A TABLE ON SQL SERVER

```
sqoop import --connect 'jdbc:sqlserver://miftah.database.windows.net:1433;
database=anirudh' --username user1 --password Pwd@123 --table abc --where
'salary>5000' --hive-import -m 1
```

```
sqoop import --connect 'jdbc:sqlserver://miftah.database.windows.net:1433;
database=anirudh' --username user1 --password Pwd@123 --query 'SELECT a.*, b.* FROM
a JOIN b on (a.id == b.id)' --hive-import -m 1
```

- **SQL QUERY EVALUATION**

```
sqoop eval --connect 'jdbc:sqlserver://miftah.database.windows.net:1433;
database=anirudh' --username user1 --password Pwd@123 --query 'SELECT * FROM tab12
LIMIT 10' -m 1
```

This is display the result of the query in the standard output.

INCREMENTAL IMPORTS IN SQOOP

Incremental import is a technique that imports only the newly added rows of a table. The following arguments control the incremental import:

- **--incremental append:** with this we can specify the column which sqoop should check to identify the new fields. Column can be specified using `--check-column` and the last value entered by `--last-value`.
- **--incremental last-modified:** This is used when the data contains a timestamp field and sqoop can use it to identify the new fields.

```
sqoop import --connect 'jdbc:sqlserver://miftah.database.windows.net:1433,
database=anirudh' --username user1 --password Pwd@123 --table table_2 --incremental
append --check-column client_id --last-value 2541 --hive-import -m 1
```

Sqoop can also automatically check for the last value in the specified column if we create Sqoop Jobs. Sqoop Job helps in rerunning the sqoop job, in case of incremental imports it automatically checks for the largest imported value of the column and imports all records with greater value.

```
sqoop-job --create job_increment --import --connect
'jdbc:sqlserver://miftah.database.windows.net:1433; database=anirudh' --username
user1 --password Pwd@123 --table table_2 --incremental append --check-column
client_id --hive-import --m 1
```

Some job commands

- **`sqoop job -list`**: to list all sqoop jobs.
- **`sqoop job --show myjob:`** to see a particular job.
- **`sqoop job --exec myjob:`** to execute the specified job.
- **`sqoop job --delete job_increment:`** to delete the job

Sqoop Link: http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html#_introduction

Some Facts

- **Azure Data Factory(ADF)** can also be used to transfer data to and from HDFS.
- Data from Local SQL server can be transferred using Gateways in ADF.
- If a Sqoop job fails in between transfer, you will have to rerun it and delete data partial transferred data yourself.

APACHE OOZIE

Schedule your jobs using oozie

Apache Oozie is a Java Web application used to **schedule Apache Hadoop jobs**. Oozie combines multiple jobs sequentially into one logical unit of work. It is integrated with the Hadoop stack, with YARN as its architectural center, and supports Hadoop jobs for Apache MapReduce, Apache Pig, Apache Hive, and Apache Sqoop.

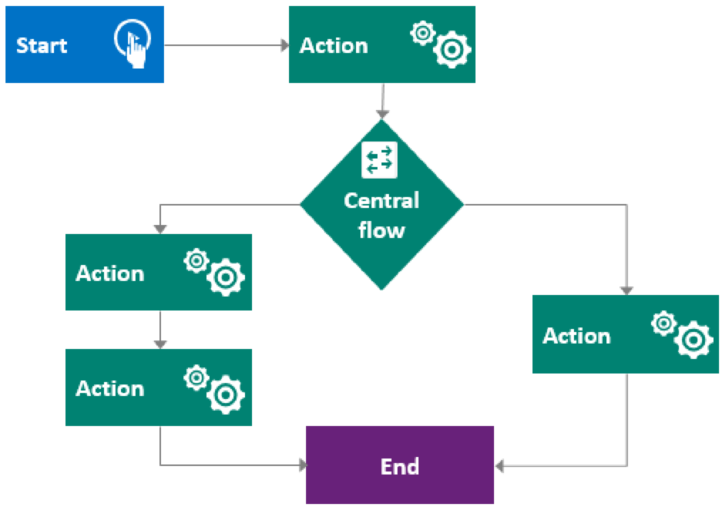
Oozie is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc.

Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique **callback HTTP URL** to the task, and notifies that URL when it is complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion

There are two basic types of Oozie jobs:

- **Oozie Workflow** jobs are Directed Acyclic Graphs (DAGs), specifying a sequence of actions to execute. The Workflow job has to wait
- **Oozie Coordinator** jobs are recurrent Oozie Workflow jobs that can be triggered by time and data availability.

Oozie Bundle provides a way to package multiple coordinator and workflow jobs and to manage the lifecycle of those jobs.

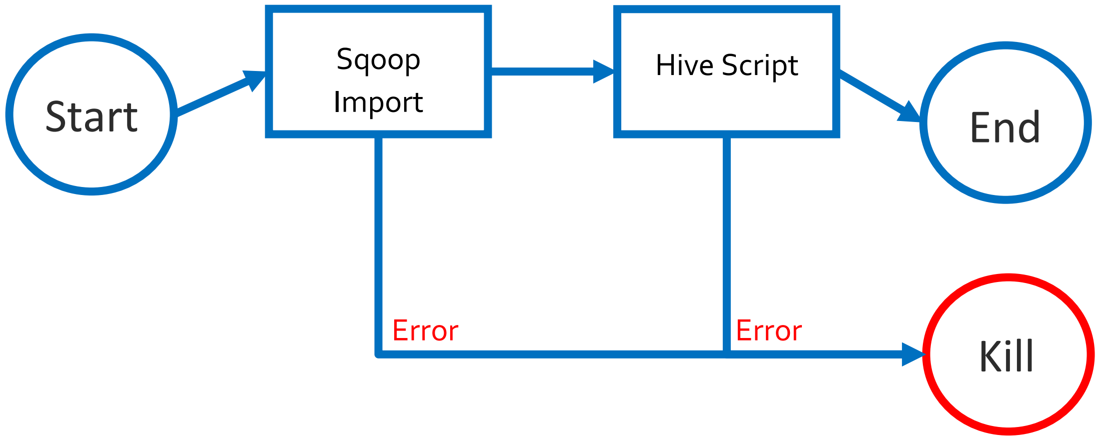


For more information: http://oozie.apache.org/docs/4.2.0/index.html

OOZIE WORKFLOW

Workflow in Oozie is a sequence of actions arranged in a control dependency DAG (Direct Acyclic Graph). The actions are in controlled dependency as the next action can only run as per the output of current action. Subsequent actions are dependent on its previous action. A workflow action can be a **Hive action, Pig action, Java action, Shell action, Sqoop action** etc.

Oozie workflows can also be parameterized (like passing ${table_name} within the workflow definition).



The workflow first imports data from the sql server and then performs some hive query on the imported data. For building such workflow we need to have the following.

- Azure CLI (use PuTTY) configured to use oozie
- Sqoop Import Details
- Hive script
- Workflow.xml (this file specifies the flow of actions)
- Job.xml (this file is used to define different variables and oozie parameters)

• CONFIGRUING PUTTY TO USE OOZIE

Once you have logged in to PuTTY, you need to configure the ssh connection to use oozie. The following will help you:

- Use the following to obtain the URL to the Oozie service:

```
sed -n '/<name>oozie.base.url/,/<\/value>/p' /etc/oozie/conf/oozie-site.xml
```

This will return a value similar to the following:

```
<name>oozie.base.url</name>
<value>http://hn0-CLUSTERNAME.cloudapp.net:11000/oozie</value>
```

The `hn0-CLUSTERNAME.cloudapp.net` is the HOSTNAME, which is also the address of JobTracker.

- Use the following to create an environment variable for the URL, so you don't have to type it for every command:

```
export OOZIE_URL=http://HOSTNAME:11000/oozie
```

- Use the following command to get the full WASB address to default storage.

  This will return a value similar to the following:

```
sed -n '/<name>fs.default/,/<\/value>/p' /etc/hadoop/conf/core-site.xml
```

```
<name>fs.defaultFS</name>
<value>wasb://mycontainer@mystorageaccount.blob.core.windows.net</value>
```

  This value will be the value of the NameNode.

- Add the user using the following, replace USERNAME with the name of user you want

```
sudo adduser USERNAME users
```

- **DEFINING HIVE QUERY**

  The query is to insert few columns of the imported table a new table. Here smalltab is the table imported from SQL server and the above query select's 3 column from it and inserts them in a different table. `${hiveTablename}` is used to define the variable. Note: Do mention the database name along with the name of table. (eg. Db.table1)

```
CREATE TABLE IF NOT EXISTS ${hiveTablename}
(cl1 int,
cl2 varchar(50),
cl3 varchar(50),
cl4 varchar(50),
cl5 int,
cl6 float,
cl7 int,
cl8 varchar(50),
cl9 int,
cl10 int);


INSERT into TABLE ${hiveTablename} Select cl1,cl2,cl5 from default.smalltab;
```

- **DEFINING THE WORKFLOW**
  This xml file consists of the actions and the order in which they need to be executed.

```
<workflow-app name="useooziewf" xmlns="uri:oozie:workflow:0.2">
    <start to = "hdfscommands"/>
    <action name="hdfscommands">
        <fs>
             <delete path="wasb://anirudh123-
      1@anirudh123.blob.core.windows.net/user/anirudh/smalltab" />
        </fs>
        <ok to="RunSqoopImport"/>
        <error to="fail"/>
    </action>

    <action name="RunSqoopImport">
        <sqoop xmlns="uri:oozie:sqoop-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.compress.map.output</name>
                    <value>true</value>
                </property>
            </configuration>
            <arg>import</arg>
            <arg>--connect</arg>
            <arg>${sqlDatabaseConnectionString}</arg>
            <arg>--table</arg>
            <arg>${sqlDatabaseTableName}</arg>
            <arg>--hive-import</arg>
            <arg>-m</arg>
            <arg>1</arg>
        </sqoop>
        <ok to="RunHiveScript"/>
        <error to="fail"/>
    </action>

    <action name="RunHiveScript">
        <hive xmlns="uri:oozie:hive-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <script>${hiveScript}</script>
            <param>hiveTablename=${hiveTablename}</param>
        </hive>
        <ok to="end"/>
        <error to="fail"/>
    </action>

    <kill name="fail">
        <message>Job failed, error
message[${wf:errorMessage(wf:lastErrorNode())}] </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

<u>Understanding some main workflow tags:</u>

- The <start-to> tag specifies the action name to begin with.
- The <ok to> tag is used to define the next action which needs to be performed if the current action is successfully executed.
- The <error to> tag tells what to do if any occurs during the execution of current action.
- The <name> tag specifies the name of the action and following this is the tag of the action type.
- The <script> tag defines the location of script file <property> tag is used to specify the value of the variables.

- **DEFINING JOB.XML**

  This is job configuration file. It contains variable values and details for the oozie job. We run this file to execute the job.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>nameNode</name>
        <value>wasb://anirudh123-1@anirudh123.blob.core.windows.net</value>
    </property>
    <property>
        <name>jobTracker</name>
        <value>hn0-
anirud.eg34vjuuc5gedm2bjlbdwv45zb.bx.internal.cloudapp.net:8050</value>
    </property>
    <property>
        <name>queueName</name>
        <value>default</value>
    </property>
    <property>
        <name>oozie.use.system.libpath</name>
        <value>true</value>
    </property>
    <property>
        <name>hiveScript</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/sqoop_oozie/useoozie/hive.hql</value>
    </property>
    <property>
        <name>hiveTablename</name>
        <value>default.test987</value>
    </property>
    <property>
        <name>sqlDatabaseConnectionString</name>
        <value>"jdbc:sqlserver://miftah.database.windows.net:1433; user=user1;
password=Pwd@123; database=anirudh"</value>
    </property>
    <property>
        <name>sqlDatabaseTableName</name>
        <value>smalltab</value>
    </property>
    <property>
        <name>hiveDataFolder</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/hive/warehouse/test123</value>
    </property>
    <property>
        <name>user.name</name>
        <value>anirudh</value>
    </property>
    <property>
        <name>oozie.wf.application.path</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/sqoop_oozie/useoozie/workflow.xml</value>
    </property>
</configuration>
```

- Once done with writing the files, move them to the blob store (use copyFromLocal). Here all the files are stored in `sqoop_oozie/useoozie` directory. Note: The job.xml file should be present on the node you are using to submit the job (the PuTTY client).
- `<name>oozie.wf.application.path</name>` This specifies the type of oozie job and the value of this property is the location of the corresponding file.
    - `oozie.wf.application.path` – For workflow
    - `oozie.coord.application.path` – For coordinator jobs
    - `oozie.bundle.application.path` – For Bundle jobs.
- Change the value corresponding to the NameNode and JobTracker.

SUBMITTING and MANAGE THE OOZIE JOB

- **Submit the Job**

```
oozie job -config job.xml -submit
```

Once the command completes, it should return the ID of the job. For example, 0000005-150622124850154-oozie-oozi-W. This will be used to manage the job.

- **Run the Job using the Job ID**

```
oozie job -start JOB-ID
```

- **Get the information about the Job**

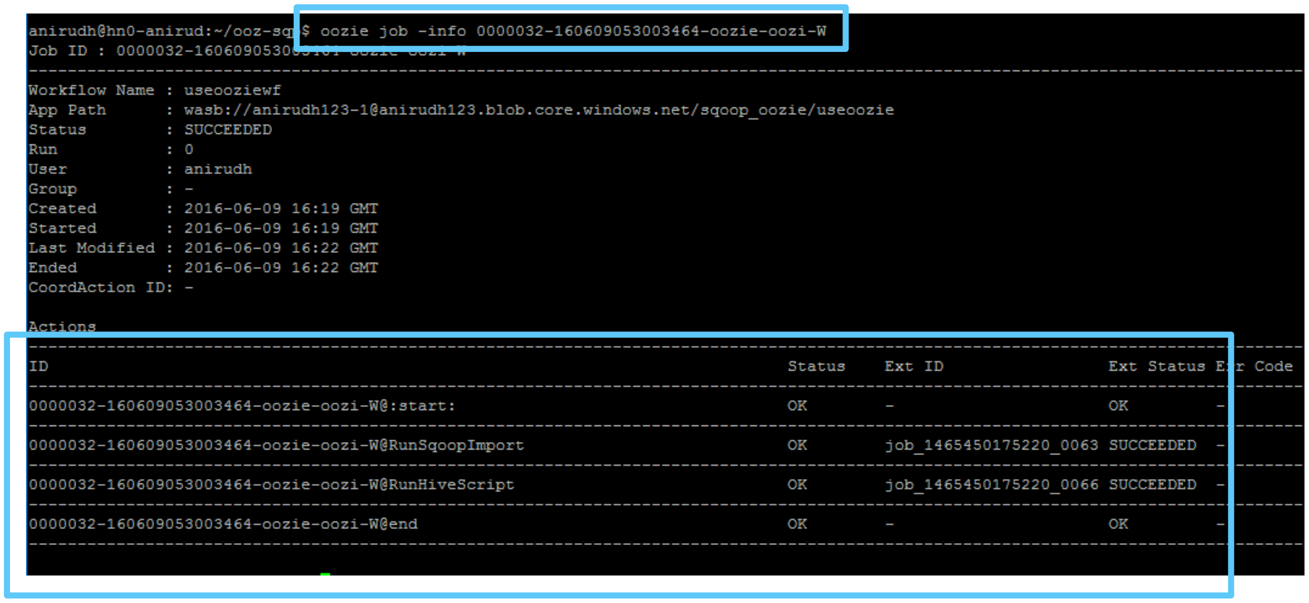
```
oozie job -info JOB-ID
```

- **See the log file of the Job**

```
oozie job -log JOB-ID
```

- **Kill a Job**

```
oozie job -kill JOB-ID
```



```
anirudh@hn0-anirud:~/ooz-sq$ oozie job -info 0000032-160609053003464-oozie-oozi-W
Job ID : 0000032-160609053003464-oozie-oozi-W
------------------------------------------------------------------------------------------------------------------------------------
Workflow Name : useooziewf
App Path      : wasb://anirudh123-1@anirudh123.blob.core.windows.net/sqoop_oozie/useoozie
Status        : SUCCEEDED
Run           : 0
User          : anirudh
Group         : -
Created       : 2016-06-09 16:19 GMT
Started       : 2016-06-09 16:19 GMT
Last Modified : 2016-06-09 16:22 GMT
Ended         : 2016-06-09 16:22 GMT
CoordAction ID: -

Actions
------------------------------------------------------------------------------------------------------------------------------------
ID                                                                             Status    Ext ID                 Ext Status Err Code
------------------------------------------------------------------------------------------------------------------------------------
0000032-160609053003464-oozie-oozi-W@:start:                                   OK        -                      OK         -
------------------------------------------------------------------------------------------------------------------------------------
0000032-160609053003464-oozie-oozi-W@RunSqoopImport                            OK        job_1465450175220_0063 SUCCEEDED  -
------------------------------------------------------------------------------------------------------------------------------------
0000032-160609053003464-oozie-oozi-W@RunHiveScript                             OK        job_1465450175220_0066 SUCCEEDED  -
------------------------------------------------------------------------------------------------------------------------------------
0000032-160609053003464-oozie-oozi-W@end                                       OK        -                      OK         -
------------------------------------------------------------------------------------------------------------------------------------
```

For more information: https://azure.microsoft.com/en-in/documentation/articles/hdinsight-use-oozie-linux-mac/#submit-and-manage-the-job

OOZIE COORDINATOR

Oozie coordinator is used to schedule the oozie workflows based on time or data availability. Creating a oozie coordinator job is very much similar to the workflows. The only thing that changes is the job.xml file and a new file coordinator.xml is added. The coordinator.xml file defines the starting time, ending time, frequency and the workflows to be executed (submitting the coordinator job starts its execution as well).

Suppose we need to run the above defined workflow every 60 mins. We will be using the above defined workflow file and create some new files to achieve this.

- ### DEFINING COORDINATATOR.XML

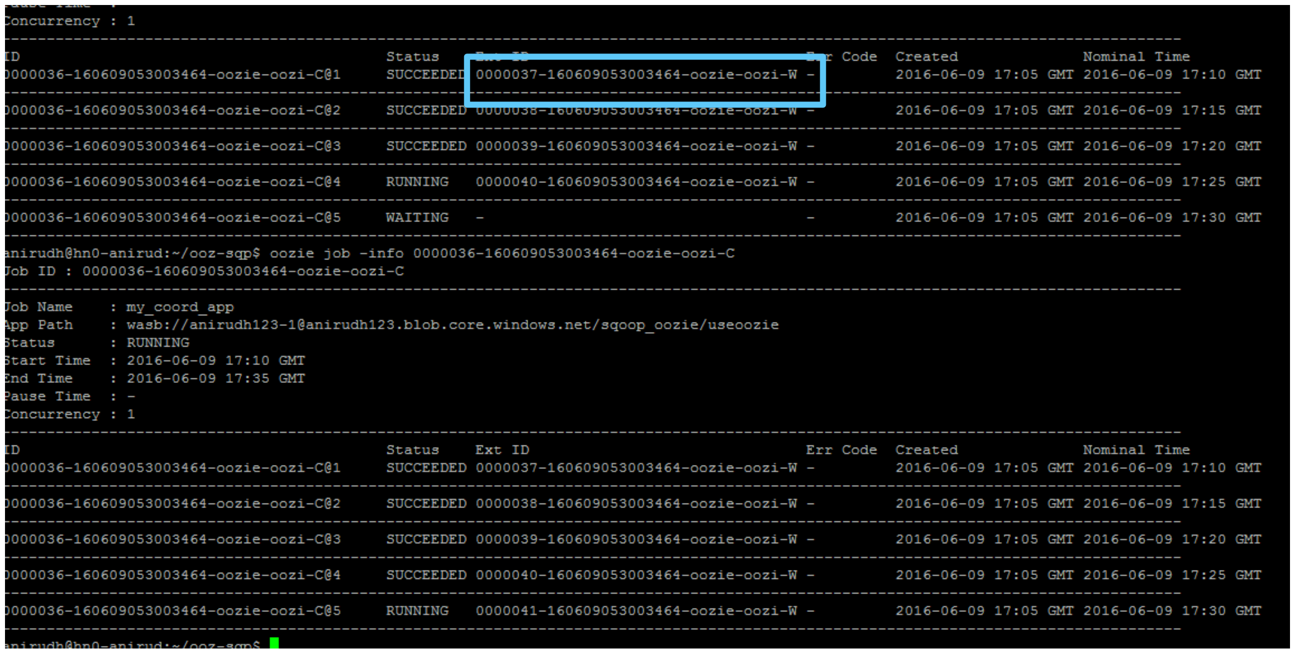
```
<coordinator-app name="my_coord_app" frequency="${coordFrequency}"
start="${coordStart}" end="${coordEnd}" timezone="${coordTimezone}"
xmlns="uri:oozie:coordinator:0.4" >

    <action>
        <workflow>
            <app-path>${wfPath}</app-path>
        </workflow>
    </action>
</coordinator-app>
```

- **DEFINING JOB.XML**

```
 <?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>nameNode</name>
        <value>wasb://anirudh123-1@anirudh123.blob.core.windows.net</value>
    </property>
    <property>
        <name>jobTracker</name>
        <value>hn0-
anirud.eg34vjuuc5gedm2bjlbdwv45zb.bx.internal.cloudapp.net:8050</value>
    </property>
    <property>
        <name>queueName</name>
        <value>default</value>
    </property>
    <property>
        <name>oozie.use.system.libpath</name>
        <value>true</value>
    </property>
    <property>
        <name>user.name</name>
        <value>anirudh</value>
    </property>
    <property>
        <name>hiveScript</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/sqoop_oozie/useoozie/hive.hql</value>
    </property>
    <property>
        <name>hiveTablename</name>
        <value>default.test987</value>
    </property>
    <property>
        <name>sqlDatabaseConnectionString</name>
        <value>"jdbc:sqlserver://miftah.database.windows.net:1433; user=user1;
password=Pwd@123; database=anirudh"</value>
    </property>
    <property>
        <name>sqlDatabaseTableName</name>
        <value>smalltab</value>
    </property>
    <property>
        <name>hiveDataFolder</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/hive/warehouse/test123</value>
    </property>
    <property>
        <name>oozie.coord.application.path</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/tutorials/useoozie</value>
    </property>
    <property>
        <name>coordFrequency</name>
        <value>60</value>
    </property>

    <property>
        <name>coordStart</name>
        <value>2016-06-09T17:10Z</value>
    </property>
    <property>
        <name>coordEnd</name>
        <value>2016-06-09T17:30Z</value>
    </property>
    <property>
        <name>coordTimezone</name>
        <value>UTC</value>
    </property>
    <property>
        <name>wfPath</name>
        <value>wasb://anirudh123-
1@anirudh123.blob.core.windows.net/tutorials/useoozie</value>
    </property>
</configuration>
```

- For debugging purpose, we can also see the info of the workflows using their JOB-ID(box).
- The frequency is defined in minutes. Some other EL time functions:

`${coord:minutes(int n)}`	`${coord:minutes(45)} --> 45 min`
`${coord:hours(int n)}`	`${coord:hours(3)} --> 180 min`
`${coord:days(int n)}`	`${coord:days(2)} --> minutes in 2 full days from the current date`
`${coord:months(int n)}`	`${coord:months(1)} --> minutes in a 1 full month from the current date`



```
Concurrency : 1
------------------------------------------------------------------------------------------------------------------------
ID                                    Status    Ext ID                               Err Code  Created              Nominal Time
0000036-160609053003464-oozie-oozi-C@1     SUCCEEDED 0000037-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:10 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@2     SUCCEEDED 0000038-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:15 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@3     SUCCEEDED 0000039-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:20 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@4     RUNNING   0000040-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:25 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@5     WAITING   -                                    -         2016-06-09 17:05 GMT 2016-06-09 17:30 GMT
------------------------------------------------------------------------------------------------------------------------
anirudh@hn0-anirud:~/ooz-sqp$ oozie job -info 0000036-160609053003464-oozie-oozi-C
Job ID : 0000036-160609053003464-oozie-oozi-C
------------------------------------------------------------------------------------------------------------------------
Job Name    : my_coord_app
App Path    : wasb://anirudh123-1@anirudh123.blob.core.windows.net/sqoop_oozie/useoozie
Status      : RUNNING
Start Time  : 2016-06-09 17:10 GMT
End Time    : 2016-06-09 17:35 GMT
Pause Time  : -
Concurrency : 1
------------------------------------------------------------------------------------------------------------------------
ID                                    Status    Ext ID                               Err Code  Created              Nominal Time
0000036-160609053003464-oozie-oozi-C@1     SUCCEEDED 0000037-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:10 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@2     SUCCEEDED 0000038-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:15 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@3     SUCCEEDED 0000039-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:20 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@4     SUCCEEDED 0000040-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:25 GMT
------------------------------------------------------------------------------------------------------------------------
0000036-160609053003464-oozie-oozi-C@5     RUNNING   0000041-160609053003464-oozie-oozi-W -         2016-06-09 17:05 GMT 2016-06-09 17:30 GMT
------------------------------------------------------------------------------------------------------------------------
anirudh@hn0-anirud:~/ooz-sqp$
```

OOZIE BUNDLE

Oozie bundles are basically used if you need to schedule coordinators at different frequencies. The **Oozie Bundle system** allows the user to define and execute a bunch of coordinator applications often called a data pipeline. There is no explicit dependency among the coordinator applications in a bundle. However, a user could use the data dependency of coordinator applications to create an implicit data application pipeline.

Note: Make sure that no 2 jobs call the same tool (like Hive) at same instance.

Scenario: We have 2 HQL files (hive1 & hive2) which ingest some data from the `hivesampletable`. The task is to schedule this ingestion between the given time duration at different frequencies. For this we will be having a workflow and a coordinator for each job, an oozie bundle that would be triggering them and a job.xml file.

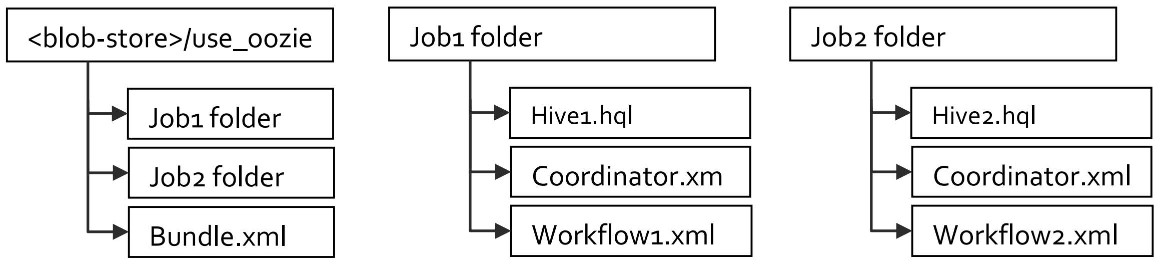
- ### DEFINING HIVE1.HQL

```
create table if not exists ${hiveTablename1}
(id string,
querytime string,
country string);

insert into ${hiveTablename1} select clientid, querytime, country from
default.hivesampletable;
```

- ### DEFINING HIVE2.HQL

```
create table if not exists ${hiveTablename2}
(id string,
state string);

insert into ${hiveTablename2} select clientid, state from default.hiveSampleTable;
```

- ### DIRECTORY STRUCTURE



- **DEFINING WORKFLOW1.XML**

```
<workflow-app name="wf1" xmlns="uri:oozie:workflow:0.2">
    <start to = "RunHiveScript"/>
    <action name="RunHiveScript">
        <hive xmlns="uri:oozie:hive-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <script>${hiveScript1}</script>
            <param>hiveTablename1=${hiveTablename1}</param>
        </hive>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Job failed, error
message[${wf:errorMessage(wf:lastErrorNode())}] </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

- **DEFINING WORKFLOW2.XML**

```
<workflow-app name="wf2" xmlns="uri:oozie:workflow:0.2">
    <start to = "RunHiveScript"/>
    <action name="RunHiveScript">
        <hive xmlns="uri:oozie:hive-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <script>${hiveScript2}</script>
            <param>hiveTablename2=${hiveTablename2}</param>
        </hive>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Job failed, error
message[${wf:errorMessage(wf:lastErrorNode())}] </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

- **DEFINING COORDINATOR.XML (1)**

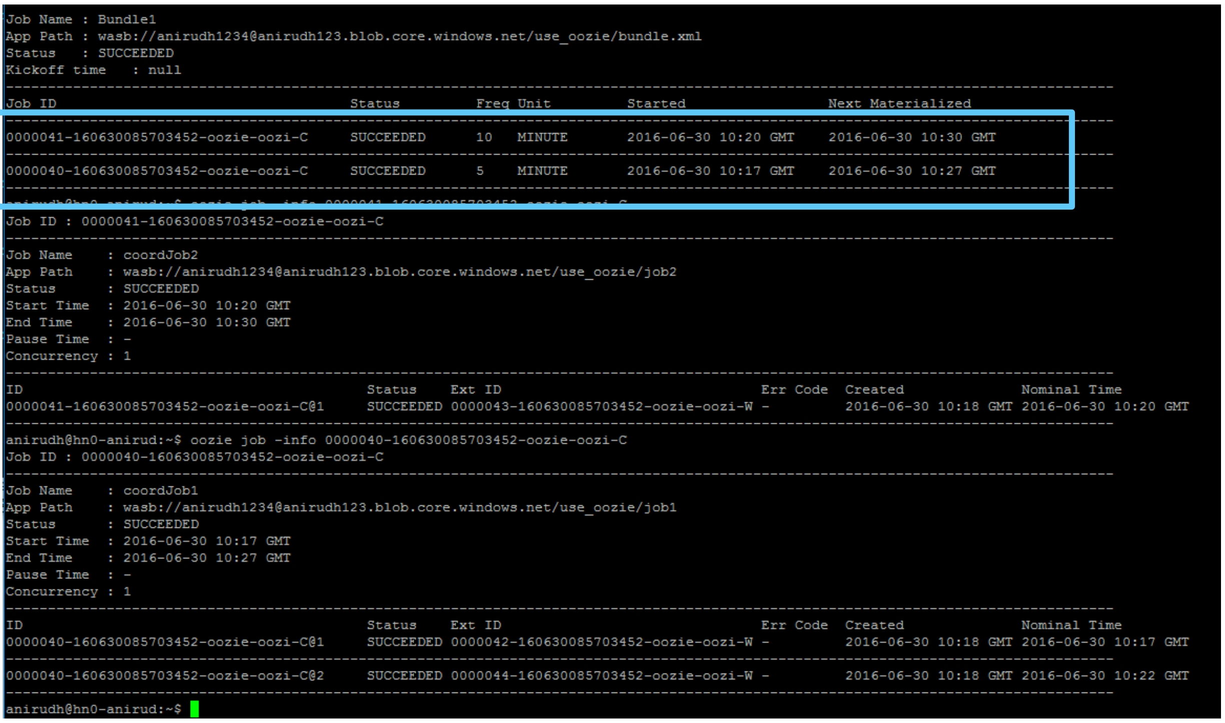
```
<coordinator-app name = "my_coord_app1" frequency = "${cf1}" start =
"${startTime1}" end = "${endTime1}"
timezone = "${coordTimezone}" xmlns = "uri:oozie:coordinator:0.4" >
   <action>
      <workflow>
         <app-path> ${wfPath1} </app-path>
      </workflow>
   </action>
</coordinator-app>
```

- **DEFINING COORDINATOR.XML (2)**

```
<coordinator-app name = "my_coord_app2" frequency = "${cf2}" start =
"${startTime2}" end = "${endTime2}"
timezone = "${coordTimezone}" xmlns = "uri:oozie:coordinator:0.4" >
   <action>
      <workflow>
         <app-path>${wfPath2}</app-path>
      </workflow>
   </action>
</coordinator-app>
```

- **DEFINING JOB.XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>nameNode</name>
        <value>wasb://anirudh1234@anirudh123.blob.core.windows.net</value>
    </property>
    <property>
        <name>jobTracker</name>
        <value>hn0-
anirud.dqj2idmzh34u5ig0qtdi4jh24f.bx.internal.cloudapp.net:8050</value>
    </property>
    <property>
        <name>queueName</name>
        <value>default</value>
    </property>
    <property>
        <name>oozie.use.system.libpath</name>
        <value>true</value>
    </property>
    <property>
        <name>hiveScript1</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job1/hive1.h
ql</value>
    </property>
    <property>
        <name>hiveScript2</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job2/hive2.h
ql</value>
    </property>
    <property>
        <name>hiveTablename1</name>
        <value>default.coord_tab_1</value>
    </property>
    <property>
        <name>hiveTablename2</name>
        <value>default.coord_tab_2</value>
    </property>
    <property>
        <name>user.name</name>
        <value>anirudh</value>
    </property>
    <property>
        <name>oozie.bundle.application.path</name>
  <value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/bundle.xml
</value>
    </property>
    <property>
        <name>coordFrequency1</name>
        <value>5</value>
    </property>
    <property>
        <name>coordFrequency2</name>
        <value>10</value>
    </property>
    <property>
        <name>START_TIME1</name>
        <value>2016-06-30T10:17Z</value>
    </property>
    <property>
        <name>START_TIME2</name>
        <value>2016-06-30T10:20Z</value>
    </property>
    <property>
        <name>END_TIME1</name>
        <value>2016-06-30T10:27Z</value>
    </property>
    <property>
        <name>END_TIME2</name>
        <value>2016-06-30T10:30Z</value>
    </property>
    <property>
        <name>coordTimezone</name>
        <value>UTC</value>
    </property>
    <property>
        <name>coordpath1</name>
        <value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job1
</value>
    </property>
    <property>
        <name>coordpath2</name>
        <value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job2
</value>
    </property>
    <property>
        <name>wfPath1</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job1/workflo
w1.xml</value>
    </property>
    <property>
        <name>wfPath2</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job2/workflo
w2.xml</value>
    </property>
</configuration>
```



```
Job Name : Bundle1
App Path : wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/bundle.xml
Status   : SUCCEEDED
Kickoff time    : null
------------------------------------------------------------------------------------------------------------------------------------
Job ID                                  Status          Freq Unit         Started                 Next Materialized
------------------------------------------------------------------------------------------------------------------------------------
0000041-160630085703452-oozie-oozi-C    SUCCEEDED       10   MINUTE       2016-06-30 10:20 GMT    2016-06-30 10:30 GMT
------------------------------------------------------------------------------------------------------------------------------------
0000040-160630085703452-oozie-oozi-C    SUCCEEDED       5    MINUTE       2016-06-30 10:17 GMT    2016-06-30 10:27 GMT
------------------------------------------------------------------------------------------------------------------------------------
anirudh@hn0-anirud:~$ oozie job -info 0000041-160630085703452-oozie-oozi-C
Job ID : 0000041-160630085703452-oozie-oozi-C
------------------------------------------------------------------------------------------------------------------------------------
Job Name    : coordJob2
App Path    : wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job2
Status      : SUCCEEDED
Start Time  : 2016-06-30 10:20 GMT
End Time    : 2016-06-30 10:30 GMT
Pause Time  : -
Concurrency : 1
------------------------------------------------------------------------------------------------------------------------------------
ID                                          Status    Ext ID                                 Err Code  Created              Nominal Time
0000041-160630085703452-oozie-oozi-C@1      SUCCEEDED 0000043-160630085703452-oozie-oozi-W -         2016-06-30 10:18 GMT 2016-06-30 10:20 GMT
------------------------------------------------------------------------------------------------------------------------------------
anirudh@hn0-anirud:~$ oozie job -info 0000040-160630085703452-oozie-oozi-C
Job ID : 0000040-160630085703452-oozie-oozi-C
------------------------------------------------------------------------------------------------------------------------------------
Job Name    : coordJob1
App Path    : wasb://anirudh1234@anirudh123.blob.core.windows.net/use_oozie/job1
Status      : SUCCEEDED
Start Time  : 2016-06-30 10:17 GMT
End Time    : 2016-06-30 10:27 GMT
Pause Time  : -
Concurrency : 1
------------------------------------------------------------------------------------------------------------------------------------
ID                                          Status    Ext ID                                 Err Code  Created              Nominal Time
0000040-160630085703452-oozie-oozi-C@1      SUCCEEDED 0000042-160630085703452-oozie-oozi-W -         2016-06-30 10:18 GMT 2016-06-30 10:17 GMT
------------------------------------------------------------------------------------------------------------------------------------
0000040-160630085703452-oozie-oozi-C@2      SUCCEEDED 0000044-160630085703452-oozie-oozi-W -         2016-06-30 10:18 GMT 2016-06-30 10:22 GMT
------------------------------------------------------------------------------------------------------------------------------------
anirudh@hn0-anirud:~$
```

TRIGGERING OOZIE JOBS BASED ON DATA AVAILABILITY

For triggering oozie jobs based on the availability of data, we can use the <datasets> provided in oozie.

A dataset is a collection of data referred to by a logical name. A dataset instance is a particular occurrence of a dataset and it is represented by a unique set of URIs. It can be individually referred. Dataset instances for datasets containing ranges are identified by a set of unique URIs, otherwise a dataset instance is identified by a single unique URI. A dataset is a synchronous (produced at regular time intervals, it has an expected frequency) input.

Syntax:

```
<dataset name="logs" frequency="${coord:days(1)}"
 initial-instance="2016-06-26T09:00Z" timezone="UTC">
    <uri-template>
      wasb://<storage_account>/log_data/${YEAR}/${MONTH}/${DAY}
    </uri-template>
    <done-flag>_SUCCESS</done-flag>
</dataset>
```

- **name:** The name of the dataset, must be a valid JAVA identifier.
- **frequency:** It represents the rate (in minutes) at which data is periodically created.
- **initial-instance:** The UTC datetime of the initial instance of the dataset.
- **uri-template:** This defines the template of the dataset path. The variables are replaced by time instance values automatically.
- **done-flag:** This specifies the name of file which tells that the data is available.

Sample Job: The inputs are available every 5 minutes in the Blob Store, the task is copy the files every 30 minutes to a new directory. We will be using pig script for this (just to get an idea of how to work with pig in oozie).

- **DEFINING PIG SCRIPT (SCRIPT_1.PIG)**

```
data = LOAD '$pigInput' USING PigStorage(',') AS
(timestamp:chararray,
ip:chararray,
browser:chararray,
type:chararray,
str:chararray,
str2:chararray);

filter1 = FILTER data BY ($2 != NULL);
STORE filter1 INTO '$pigOutput' using PigStorage(',');
```

- **DEFINING WORKFLOW.XML**

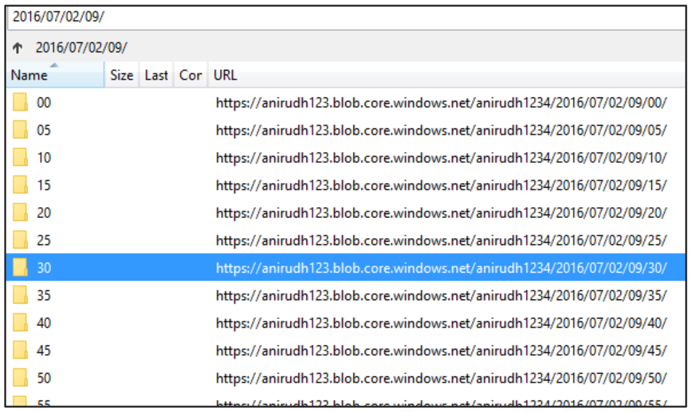
```
<workflow-app name="wf2" xmlns="uri:oozie:workflow:0.2">
    <start to = "RunPig"/>
    <action name="RunPig">
        <pig>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <prepare>
                <delete path="${pigOutput}"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.compress.map.output</name>
                    <value>true</value>
                </property>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <script>${pigscript1}</script>
            <param>pigInput=${pigInput}</param>
            <param>pigOutput=${pigOutput}</param>
        </pig>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Job failed, error
message[${wf:errorMessage(wf:lastErrorNode())}] </message>
    </kill>
    <end name="end"/>
</workflow-app>
```

- **DEFINING COORDINATOR.XML**

```
<coordinator-app name = "my_coord_app2" frequency = "${cf1}" start =
"${startTime2}" end = "${endTime2}"
  timezone = "${coordTimezone}" xmlns = "uri:oozie:coordinator:0.4" >


    <datasets>
       <dataset name="logs" frequency="${cf2}" initial-
       instance="${initial_instance}" timezone="UTC">
          <uri-template>
          wasb://anirudh1234@anirudh123.blob.core.windows.net/${YEAR}/${MONTH}/${
          DAY}/${HOUR}/${MINUTE}</uri-template>
         <done-flag>_SUCCESS</done-flag>
       </dataset>

       <dataset name="pig_out" frequency="${cf2}" initial-
       instance="${initial_instance}" timezone="${coordTimezone}">
            <uri-template>
            wasb://anirudh1234@anirudh123.blob.core.windows.net/pig_output/${YEAR
            }-${MONTH}-${DAY}-${HOUR}-${MINUTE}</uri-template>
       </dataset>
   </datasets>

    <input-events>
        <data-in name="input" dataset="logs">
            <start-instance>${coord:current(-6)} </start-instance>
            <end-instance>${coord:current(-1)}</end-instance>
        </data-in>
    </input-events>

    <output-events>
        <data-out name="output" dataset="pig_out">
            <instance>${coord:current(0)}</instance>
        </data-out>
    </output-events>

    <action>
        <workflow>
            <app-path>${wfPath2}</app-path>
            <configuration>
                <property>
                    <name>pigInput</name>
                    <value>${coord:dataIn('input')}</value>
                </property>
                <property>
                    <name>pigOutput</name>
                    <value>${coord:dataOut('output')}</value>
                </property>
            </configuration>
        </workflow>
    </action>
</coordinator-app>
```

- **DEFINING JOB.XML**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>nameNode</name>
        <value>wasb://anirudh1234@anirudh123.blob.core.windows.net</value>
    </property>
    <property>
        <name>jobTracker</name>
        <value>hn0-
anirud.uzh4cjfjulmenmolf02j2bfkld.bx.internal.cloudapp.net::8050</value>
    </property>
    <property>
        <name>queueName</name>
        <value>default</value>
    </property>
    <property>
        <name>oozie.use.system.libpath</name>
        <value>true</value>
    </property>
    <property>
        <name>pigscript1</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/oozie_time/job1/pig1.p
ig</value>
    </property>
    <property>
        <name>user.name</name>
        <value>anirudh</value>
    </property>
    <property>
        <name>oozie.coord.application.path</name>
<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/oozie_time/job1/coordi
nator.xml</value>
    </property>
    <property>
        <name>cf1</name>
        <value>30</value>
    </property>
    <property>
        <name>cf2</name>
        <value>5</value>
    </property>
    <property>
        <name>startTime2</name>
        <value>2016-07-02T07:30Z</value>
    </property>
    <property>
        <name>endTime2</name>
        <value>2016-07-02T010:30Z</value>
    </property>
    <property>
        <name>initial_instance</name>
        <value>2016-07-02T07:00Z</value>
    </property>
    <property>
        <name>wfPath2</name>

<value>wasb://anirudh1234@anirudh123.blob.core.windows.net/oozie_time/job1/workfl
ow1.xml</value>
    </property>
</configuration>
```

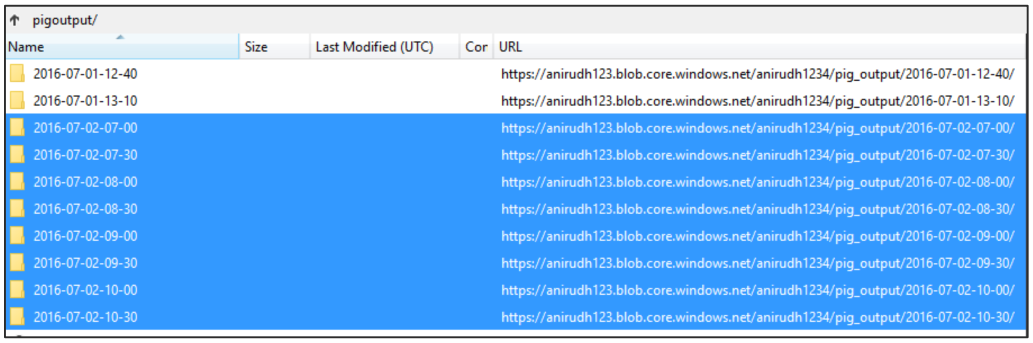


2016/07/02/09/

↑ 2016/07/02/09/

Name	Size	Last	Cor	URL
00				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/00/
05				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/05/
10				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/10/
15				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/15/
20				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/20/
25				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/25/
30				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/30/
35				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/35/
40				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/40/
45				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/45/
50				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/50/
55				https://anirudh123.blob.core.windows.net/anirudh1234/2016/07/02/09/55/

Fig: Input Files



↑ pigoutput/

Name	Size	Last Modified (UTC)	Cor	URL
2016-07-01-12-40				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-01-12-40/
2016-07-01-13-10				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-01-13-10/
2016-07-02-07-00				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-07-00/
2016-07-02-07-30				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-07-30/
2016-07-02-08-00				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-08-00/
2016-07-02-08-30				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-08-30/
2016-07-02-09-00				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-09-00/
2016-07-02-09-30				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-09-30/
2016-07-02-10-00				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-10-00/
2016-07-02-10-30				https://anirudh123.blob.core.windows.net/anirudh1234/pig_output/2016-07-02-10-30/

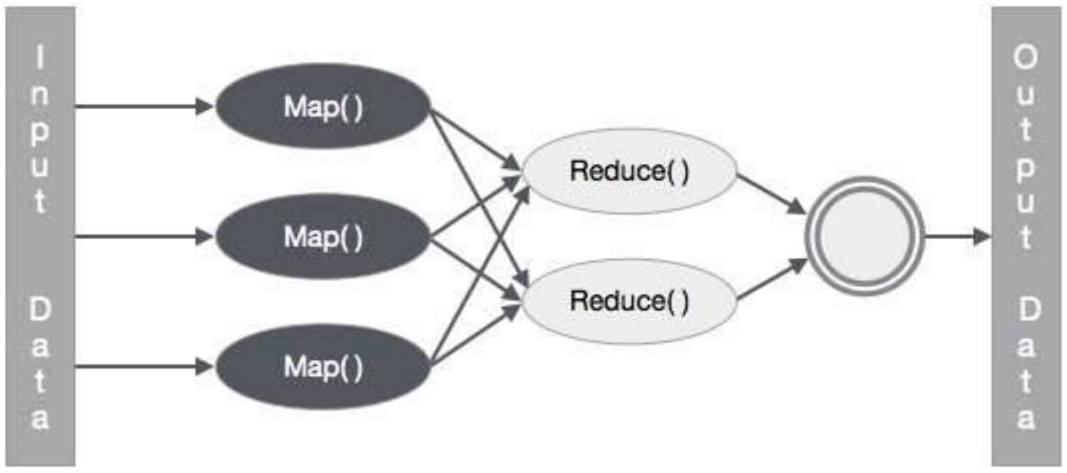
Fig: Output Files

For more information on datasets read from page no. 118 Book: Apache Oozie by O'Reilly
Other Links:
https://oozie.apache.org/docs/3.1.3-incubating/CoordinatorFunctionalSpec.html#a5._Dataset
http://www.tutorialspoint.com/apache_oozie/apache_oozie_introduction.html
https://azure.microsoft.com/en-in/documentation/articles/hdinsight-use-oozie-linux-mac/

MAPREDUCE JOBS ON HDINSIGHT

HDInsight allows you to submit your own reduce jobs. Hadoop MapReduce is a software framework for writing jobs that process vast amounts of data. Input data is split into independent chunks, which are then processed in parallel across the nodes in your cluster. A MapReduce job consist of two functions:

- **Mapper**: Consumes input data, analyzes it (usually with filter and sorting operations), and emits tuples (key-value pairs)
- **Reducer**: Consumes tuples emitted by the Mapper and performs a summary operation that creates a smaller, combined result from the Mapper data



For running a mapreduce job we create an apache Maven project and write our java code in that and create the dependencies. Building this package gives the .jar file which we upload on the ssh client and execute it using on our cluster with the path of input and output files.

Following steps helps to submit a custom mapreduce job to find the maximum value of a column.

- **Install Apache Maven-** Download it and install it. You may need to set the path variables in the System Settings.
- **Create a Maven Project-** Open command prompt and move to the directory where you want to create the project. Use the following to create the project (new123 is the project name)

```
mvn archetype:generate -DgroupId=org.apache.hadoop.examples -
DartifactId=new123 -DarchetypeArtifactId=maven-archetype-quickstart -
DinteractiveMode=false
```

- You will notice a project folder. Remove the AppTest.java file from new123/test/...../apptest.java
- Open the pom file and add the following just above the </dependencies> tag.

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-examples</artifactId>
    <version>2.5.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-common</artifactId>
    <version>2.5.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>2.5.1</version>
    <scope>provided</scope>
</dependency>
```

- Add the following in between the </dependencies> and </project> tag

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>2.3</version>
            <configuration>
                <transformers>
                    <transformer
implementation="org.apache.maven.plugins.shade.resource.ApacheLicenseResource
Transformer">
                    </transformer>
                </transformers>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.7</source>
                <target>1.7</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

- Put your mapreducer code in the src/main/java/org....../FindMax.java.

```
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.Iterator;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class FindMax{

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable> {

        private Text word = new Text();
        private IntWritable res = new IntWritable();
        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
            String line = value.toString();
            String[] s = line.split(",");
            word.set(s[0]);
            int l = s.length;
            //System.out.println(s[0] + "    " + l);
            int i = 0;
            int sum;
            for(i = 1; i < l; i++) {
                sum = Integer.parseInt(s[i]);
                res.set(sum);
                context.write(word, res);
            }
        }
    }
    public static class IntSumReducer
            extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        @Override
        protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {

            int max = 0;
            int val2 = 0;
            for (IntWritable value : values) {
                val2 = value.get();
                if(val2 > max)
                   max = val2;
            }
            result.set(val2);
            context.write(key, result);
        }

    }
```

```
  public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, 
args).getRemainingArgs();
        if (otherArgs.length != 2) {
            System.err.println("Usage: <in> <out>");
            System.exit(2);
        }
        Job job = new Job(conf, "word count");
        job.setJarByClass(FindMax.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

- Move to the Maven folder directory in command prompt and build the package using

```
mvn clean package
```

- In the target folder of the project directory you will find a jar file for the project, name would be like `new123-1.0-SNAPSHOT.jar`
- Upload the file on the NameNode using **winscp** software.
- Once you have uploaded the jar file use the following to submit it.

```
yarn jar new123-1.0-SNAPSHOT.jar org.apache.hadoop.examples.FindMax 
wasb:///data.csv wasb:///map_red_output/1
```

  - Here `new123-1.0-SNAPSHOT.jar` is the name of jar file
  - `FindMax` is the name of the class
  - `wasb:///data.csv` is the input data file
  - `wasb:///map_red_output/1` is the file directory to store the output

UNDERSTANDTING THE MAPREDUCE JAVA CODE

There are 3 basic functions in the java class

- **Mapper Function** - It breaks the data into key-value pairs. The function receives the first 2 parameters and defines the output key and value (the other 2 parameters) using them.
  - Parameters -
    - Input key
    - The input value (the Row of data file)
    - The output key

                - The value corresponding to key
        - A key can have multiple values (value can aslo be a class object)
        - Context here refers to the output parameters.
        - The output (output key and value pair) of this is the input of the Reducer.
- **Reducer Function-** It receives all key value pairs having same key at a time
    - Parameters- Similar to the map function. The output of mapper is the input here.
    - Computations are performed here the final output is calculated.
- **Main() Function-**
    - This is the driver function.
    - It Specifies the mapper, reducer and other class names along with the type of output key and values.
    - It also specifies how to deal with the arguments for input & output.

E-Book (for examples):

- http://www.nataraz.in/data/ebook/hadoop/mapreduce_design_patterns.pdf

Links:

- http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Purpose
- https://azure.microsoft.com/en-in/documentation/articles/hdinsight-use-mapreduce/
- https://azure.microsoft.com/en-in/documentation/articles/hdinsight-develop-deploy-java-mapreduce-linux/
- http://www.tutorialspoint.com/map_reduce/implementation_in_hadoop.htm
- https://www.packtpub.com/books/content/understanding-mapreduce