

C o m m u n i t y E x p e r i e n c e D i s t i l l e d

HDInsight Essentials

Tap your unstructured Big Data and empower your business
using the Hadoop distribution from Windows

Rajesh Nadipalli

[PACKT]
PUBLISHING

HDInsight Essentials

Tap your unstructured Big Data and empower your business using the Hadoop distribution from Windows

Rajesh Nadipalli



BIRMINGHAM - MUMBAI

HDInsight Essentials

Copyright © 2013 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2013

Production Reference: 1160913

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84969-536-7

www.packtpub.com

Cover Image by Suresh Mogre (suresh.mogre.99@gmail.com)

Credits

Author

Rajesh Nadipalli

Project Coordinator

Sherin Padayatty

Reviewers

Anindita Basak

Avkash Chauhan

Durga Ramana Muktevi

Proofreader

Hardip Sidhu

Indexer

Monica Ajmera Mehta

Acquisition Editor

Sam Birch

Erol Staveley

Graphics

Yuvraj Mannari

Valentina Dsilva

Disha Haria

Commissioning Editor

Mohammed Fahad

Production Coordinator

Adonia Jones

Technical Editors

Dennis John

Gaurav Thingalaya

Cover Work

Adonia Jones

Copy Editor

Tanvi Gaitonde

Sayanee Mukherjee

Adithi Shetty

Laxmi Subramanian

About the Author

Rajesh Nadipalli has over 17 years' IT experience and has held a technical leadership position at Cisco Systems. His key focus areas have been Data Management, Enterprise Architecture, Business Intelligence, Data Warehousing, and Extract Transform Load (ETL). He has demonstrated success by delivering scalable data management and BI solutions that empower businesses to make informed decisions.

In his current role as a Senior Solutions Architect at Zaloni, Raj evaluates Big Data goals for his clients, recommends a target state architecture, assists in proof of concepts, and prepares them for a production implementation. In addition, Raj is an instructor for Hadoop for Developers, Hive, Pig, and HBase. His clients include Verizon, American Express, Netapp, Cisco, EMC, and United Health Group.

Raj holds an MBA from NC State University and a BS in EE from the University of Mumbai, India.

I would like to thank my family.

About the Reviewers

Anindita Basak is currently working as a Senior System Analyst in Sonata Software in the Windows Azure Delivery group of Microsoft. She has worked as a Senior Software Engineer on implementations of various enterprise applications on Windows Azure and Windows phones. She started her journey in Windows Azure in the Microsoft Cloud Integration Engineering (CIE) team and worked as a Support Engineer in Microsoft India (R&D) Pvt. Ltd. With more than 5 years' experience in the Microsoft .NET technology stack, she is solely focused on Cloud and Microsoft Mobility. She loves to share her technical experience and expertise through her blog at <http://anindita9.wordpress.com>.

I would like to thank my parents and my lovely brother! Without your help I can't achieve any goals of my life.

Avkash Chauhan has spent the last 8 years at Microsoft where he was a founding member of the HDInsight product, and has recently started working at a start-up named Platfora Inc. as a Senior Engineer. Avkash Chauhan worked on the HDInsight project since its incubation at Microsoft, and worked with a few early stage customers at Microsoft. Avkash Chauhan has spent about five years in cloud computing, and for the last 3 years, he has been working with Hadoop technology developing products to make Hadoop amazing.

I would like to thank my family and friends, who gave me their love and time to review this book.

Durga Ramana Muktevi has 16 years' IT experience and held various positions in implementing as well as managing enterprise scale Information Technology architectures. He has domain expertise in banks, and also in financial, health, and insurance industries. He has worked for Fortune 500, start-ups, and the Big Four consulting companies.

He has held various positions in management, IT strategy, M&A, and large transformations. His architectural and technological experience spans wide and broadly in the disciplines such as enterprise, business, application, data, security, operations, infrastructure, cloud, SaaS, Big Data, ERP, SOA, modernization, product development, and integration.

He has held responsible positions for governance and supporting government regulatory and compliance, as well as internal and external audits.

Durga Muktevi has a Bachelors degree in Electronics and Communications Engineering from Karnataka University, Dharwad, India. He has attended various conferences, technology, and business administration classes. He is married, blessed with two kids, and lives in Cary, North Carolina, USA.

He has worked at McKesson, Royal Bank of Canada, QSecure, and Unitrin Direct. He has also worked as a consultant for Accenture, Fujitsu Consulting, and SV Consulting.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Hadoop and HDInsight in a Heartbeat	5
Big Data – hype or real?	5
Apache Hadoop concepts	6
Core components	7
Hadoop cluster layout	8
The Hadoop ecosystem	9
Data access	9
Data processing	10
The Hadoop data store	10
Management and integration	10
Hadoop distributions	10
HDInsight distribution differentiator	11
End-to-end solution using HDInsight	11
Key phases of a Hadoop project	12
Summary	22
Chapter 2: Deploying HDInsight on Premise	23
HDInsight and Hadoop relationship	23
Deployment options for on-premise	24
Windows HDInsight server	24
Hortonworks Data Platform (HDP for Windows)	25
Supported platforms for on-premise install	25
Single-node install	25
Downloading the software	25
Running the install wizard	26
Validating the install	26
Multinode planning and preparation	27
Setting up the network	28
Setting common time on all nodes	29

Setting up remote scripting	29
Configuring firewall ports	31
Multinode installation	32
Downloading the software	32
Configuring the multinode install	32
Running the installer	33
Validating the install	33
Managing HDInsight services	34
Uninstalling HDInsight	34
Summary	35
Chapter 3: HDInsight Azure Cloud Service	37
HDInsight Service on Azure	37
Considerations for Azure HDInsight Service	38
Provision your cluster	38
HDInsight management dashboard	40
Verify the cluster and run sample jobs	41
Access HDFS	41
Deploy and execute the sample MapReduce job	42
View job results	43
Monitor your cluster	44
Azure storage integration	45
Remove your cluster	46
Delete your cluster	46
Delete your storage	47
Restore your cluster	47
Summary	48
Chapter 4: Administering Your HDInsight Cluster	49
Cluster status	50
Distributed filesystem health	50
NameNode URL	50
Browsing HDFS	51
MapReduce health	51
MapReduce summary	52
MapReduce Job History	52
Key files	53
Backing up NameNode content	53
Summary	54

Chapter 5: Ingesting Data to Your Cluster	55
Loading data using Hadoop commands	55
Step 1 – connect to a Hadoop client	55
Step 2 – get your files on local storage	56
Step 3 – upload to HDFS	57
Loading data using Azure Storage Vault (ASV)	57
Storage access keys	57
Storage tools	58
Azure Storage Explorer	58
Registering your storage account	59
Uploading files to your blob storage	59
Loading data using interactive JavaScript	60
Shipping data to Azure	61
Loading data using Sqoop	62
Key benefits	62
Two modes of using Sqoop	62
Using Sqoop to import (SQL to Hadoop)	63
Summary	64
Chapter 6: Transforming Data in Cluster	65
Transformation scenario	65
Scenario	65
Transformation objective	66
File organization	66
MapReduce solution	67
Design	67
Map code	67
Reduce code	69
Driver code	70
Compiling and packaging the code	71
Executing MapReduce	71
Results verification	71
Hive solution	72
Overview of Hive	72
Starting Hive in the HDInsight node	73
Step 1 – table creation	74
Step 2 – table loading	75
Step 3 – summary table creation	75
Step 4 – verifying the summary table	76
Pig solution	77
Pig architecture	77
Pig or Hive?	78

Table of Contents

Starting Pig in the HDInsight node	78
Pig Grunt script	78
Code	79
Code explanation	79
Execution	80
Verification	81
Summary	82
Chapter 7: Analyzing and Reporting Your Data	83
Analyzing and reporting using Excel	83
Step 1 – installing the Hive ODBC driver	84
Step 2 – creating Hive ODBC data source	84
Step 3 – importing data to Excel	86
Hive for ad hoc queries	90
Creating reference tables	90
Ad hoc queries	91
Analytic functions in HiveQL	92
Interactive JavaScript for analysis and reporting	92
Other business intelligence tools	92
Summary	93
Chapter 8: Project Planning Tips and Resources	95
Architectural considerations	95
Extensible and modular	95
Metadata-driven solution	96
Integration strategy	96
Security	96
Project planning	97
Proof of Concept	97
Production implementation	97
Reference sites and blogs	98
Summary	99
Index	101

Preface

This book gives a quick introduction to Hadoop-like problems, and gives a primer on the real value of HDInsight. Next, it will show how to set up your HDInsight cluster. Then, it will take you through the four stages: collect, process, analyze, and report. For each of these stages you will see a practical example with the working code.

What this book covers

Chapter 1, Hadoop and HDInsight in a Heartbeat, provides an overview of Hadoop and Microsoft HDInsight distribution. We will review core Hadoop concepts and get a primer on what's involved to build an end-to-end solution using HDInsight.

Chapter 2, Deploying HDInsight on Premise, uncovers the links between HDInsight, Hadoop, and Hortonworks data platform. We will then install and set up HDInsight on the single-node and multinode cluster. We will verify the installation and review the uninstallation procedure.

Chapter 3, HDInsight Azure Cloud Service, registers and provisions a HDInsight using the Azure cloud service. We will verify the installation and tear it down by preserving data in the Azure storage vault with the option to restore.

Chapter 4, Administer Your HDInsight Cluster, explains the common administration responsibilities including cluster status, NameNode status, JobTracker status, and the key file locations.

Chapter 5, Ingest Data to Your Cluster, covers various methods to load data to your cluster via Hadoop commands, Azure storage vault, and Sqoop.

Chapter 6, Transform Data in Cluster, covers a real-life Hadoop transformation need and how to use MapReduce, Hive, and Pig to solve this need.

Chapter 7, Analyze and Report Your Data, explains how to analyze and report using Excel, ad hoc queries using Hive, interactive JavaScript, and other Business Intelligence tools.

Chapter 8, Project Planning Tips and Resources, reviews architectural considerations, project planning, and reference sites.

What you need for this book

To run the sample programs, you need to have a Windows Azure account and request the HDInsight preview service or install the software on your site. Supported platforms for on-premise installation are given below:

Type	Supported operating system
Single node	Windows 8, Windows 7, Vista SP2, XP SP3+, Server 2003 SP2+, Server 2008, and Server 2008 R2
HDP for Windows	Windows 8, Windows 7, Server 2008 R2, and Server 2012

Who this book is for

This book is designed for data architects, developers, and managers who want to understand Hadoop and leverage Windows infrastructure. It is ideal for teams who are starting/planning a Hadoop implementation and want to understand key Hadoop components and how HDInsight provides value addition in development, administration, and reporting.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "We can include other contexts through the use of the `include` directive."


A block of code is set as follows:


```
public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.out.println("Usage: WordCount <input dir> <output
dir>");
        System.exit(-1);
    }
}
```

Any command-line input or output is written as follows:

```
C:\Users\rajn\SampleCode\WordCount>hadoop jar WordCount.jar WordCount /
user/hadoop/wcount/Nasa_Human_Spaceflight_Report.txt /user/hadoop/wcount/
NasaWordCountOutput
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "clicking the **Next** button moves you to the next screen".

[ Warnings or important notes appear in a box like this.]

[ Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Hadoop and HDInsight in a Heartbeat

Apache Hadoop is the trademark of Apache Software Foundation (<http://hadoop.apache.org/>). HDInsight is Microsoft's Apache Hadoop-based solution for Windows server and Windows Azure. This chapter will provide an overview of Apache Hadoop and Microsoft Big Data strategy where Microsoft HDInsight plays an important role. We will cover the following topics:

- Big Data: hype or real?
- Hadoop core concepts
- End-to-end Big Data solutions using HDInsight

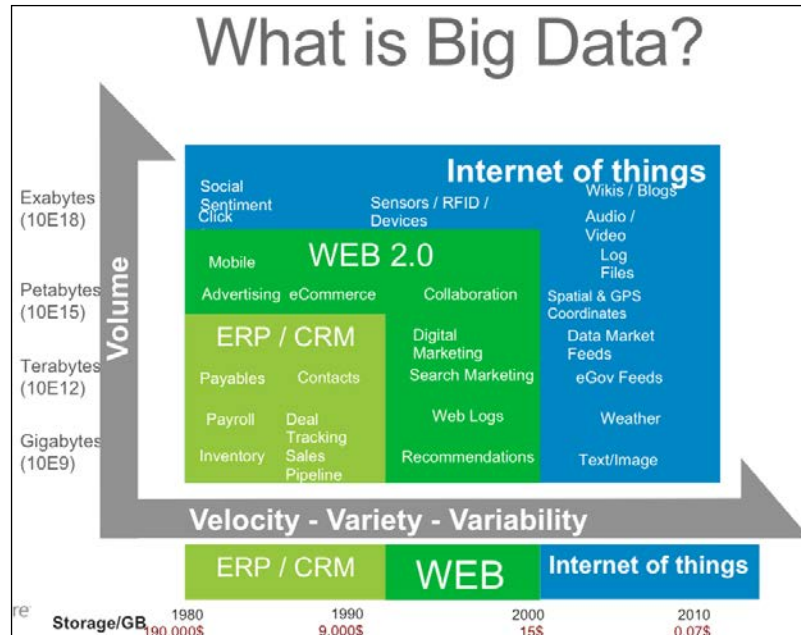
Big Data – hype or real?

We live in an era where data is generated with every action. Following are some interesting facts based on the 2012 social-media data: 750 tweets per second, 575 Instagram likes every second, 5 Billion Google+ button clicks per day, and 130 friends per average Facebook user (Source: <http://www.go-globe.com/blog/social-media-facts/>). Big Data is not hype but a reality.

Some more Big Data use cases to demystify any skeptics are as follows:

- Utility sensors that tracked electricity consumption for a small town a decade ago were manual, and information was managed in a relational database. Today, these are digital and able to collect millions of data points per minute. Older relational databases cannot sustain this growth.
- Latest medical sensors allow asthma patients to be better prepared by alerting them, if weather data in an area is forecasted to be harsh. The systems now track who needs care and when they need care based on various data feeds.

- The following screenshot shows a trend analysis done by Microsoft Azure that shows volume across the three key platforms: ERP, Web 2.0, and Internet of things. The section in blue highlights the Big Data challenge where **Volume**, **Velocity**, and **Variety** are high like Wikis, Video, Audio, and Sensors.



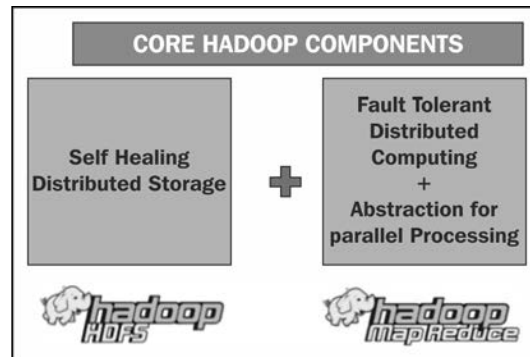
The image source is as follows: <http://anindita9.files.wordpress.com/2013/06/bigdata.jpg>.

Apache Hadoop concepts

Apache Hadoop is the leading Big Data platform that allows to process large datasets efficiently and at low cost. Other Big Data platforms are MongoDB, Cassandra, and CouchDB. This section describes Apache Hadoop core concepts and its ecosystem.

Core components

The following image shows core Hadoop components:



At the core, Hadoop has two key components:

- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce (distributed computing for batch jobs)

For example, say we need to store a large file of 1 TB in size and we only have some commodity servers each with limited storage. Hadoop Distributed File System can help here. We first install Hadoop, then we import the file, which gets split into several blocks that get distributed across all the nodes. Each block is replicated to ensure that there is redundancy. Now we are able to store and retrieve the 1 TB file.

Now that we are able to save the large file, the next obvious need would be to process this large file and get something useful out of it, like a summary report. To process such a large file would be difficult and/or slow if handled sequentially. Hadoop MapReduce was designed to address this exact problem statement and process data in parallel fashion across several machines in a fault-tolerant mode. MapReduce programming models use simple key-value pairs for computation.

One distinct feature of Hadoop in comparison to other cluster or grid solutions is that Hadoop relies on the "share nothing" architecture. This means when the MapReduce program runs, it will use the data local to the node, thereby reducing network I/O and improving performance. Another way to look at this is when running MapReduce, we bring the code to the location where the data resides. So the code moves and not the data.

HDFS and MapReduce together make a powerful combination, and is the reason why there is so much interest and momentum with the Hadoop project.

Hadoop cluster layout

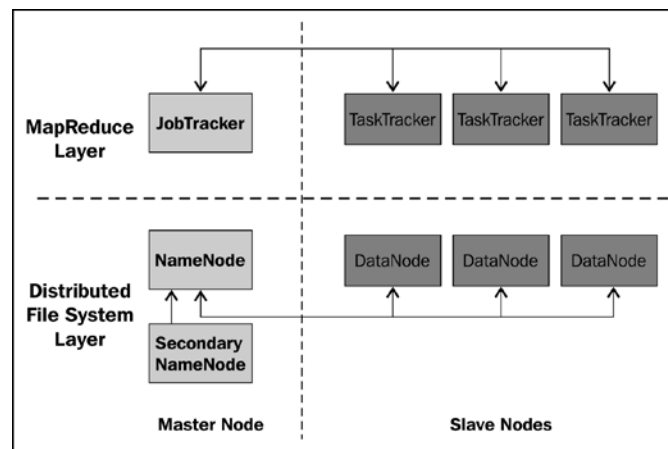
Each Hadoop cluster has three special master nodes (also known as servers):

- **NameNode:** This is the master for the distributed filesystem and maintains a metadata. This metadata has the listing of all the files and the location of each block of a file, which are stored across the various slaves (worker bees). Without a NameNode HDFS is not accessible.
- **Secondary NameNode:** This is an assistant to the NameNode. It communicates only with the NameNode to take snapshots of the HDFS metadata at intervals configured at cluster level.
- **JobTracker:** This is the master node for Hadoop MapReduce. It determines the execution plan of the MapReduce program, assigns it to various nodes, monitors all tasks, and ensures that the job is completed by automatically relaunching any task that fails.

All other nodes of the Hadoop cluster are slaves and perform the following two functions:

- **DataNode:** Each node will host several chunks of files known as blocks. It communicates with the NameNode.
- **TaskTracker:** Each node will also serve as a slave to the JobTracker by performing a portion of the map or reduce task, as decided by the JobTracker.

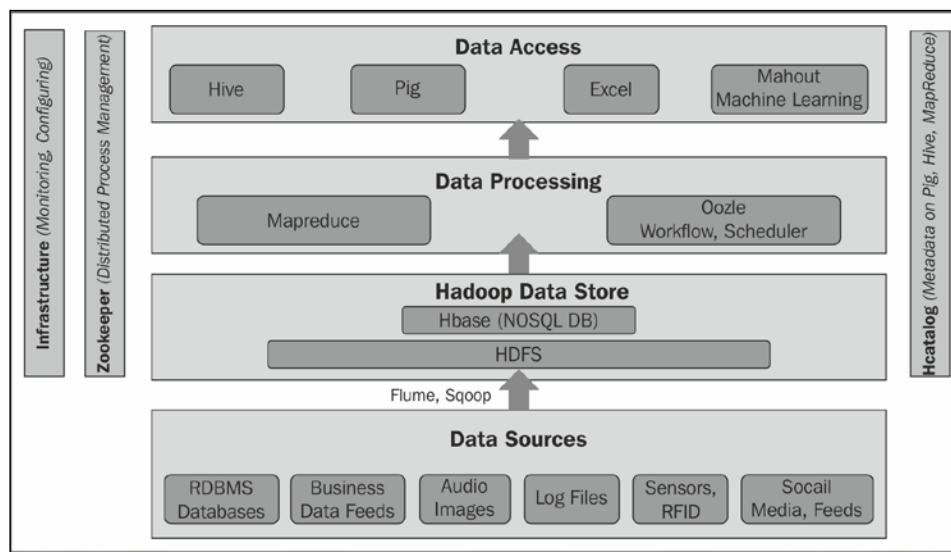
The following image shows a typical Apache Hadoop cluster:



The Hadoop ecosystem

As Hadoop's popularity has increased, several related projects have been created that simplify accessibility and manageability to Hadoop. I have organized them as per the stack, from top to bottom.

The following image shows the Hadoop ecosystem:



Data access

The following software are typically used access mechanisms for Hadoop:

- **Hive:** It is a data warehouse infrastructure that provides SQL-like access on HDFS. This is suitable for the ad hoc queries that abstract MapReduce.
- **Pig:** It is a scripting language such as Python that abstracts MapReduce and is useful for data scientists.
- **Mahout:** It is used to build machine learning and recommendation engines.
- **MS Excel 2013:** With HDInsight, you can connect Excel to HDFS via Hive queries to analyze your data.

Data processing

The following are the key programming tools available for processing data in Hadoop:

- MapReduce: This is the Hadoop core component that allows distributed computation across all the TaskTrackers
- Oozie: It enables creation of workflow jobs to orchestrate Hive, Pig, and MapReduce tasks

The Hadoop data store

The following are the common data stores in Hadoop:

- HBase: It is the distributed and scalable NOSQL (Not only SQL) database that provides a low-latency option that can handle unstructured data
- HDFS: It is a Hadoop core component, which is the foundational distributed filesystem

Management and integration

The following are the management and integration software:

- Zookeeper: It is a high-performance coordination service for distributed applications to ensure high availability
- Hcatalog: It provides abstraction and interoperability across various data processing software such as Pig, MapReduce, and Hive
- Flume: Flume is distributed and reliable software for collecting data from various sources for Hadoop
- Sqoop: It is designed for transferring data between HDFS and any RDBMS

Hadoop distributions

Apache Hadoop is an open-source software and is repackaged and distributed by vendors offering enterprise support. The following is the listing of popular distributions:

- Amazon Elastic MapReduce (cloud, <http://aws.amazon.com/elasticmapreduce/>)
- Cloudera (<http://www.cloudera.com/content/cloudera/en/home.html>)

- EMC PivotalHD (<http://gopivotal.com/>)
- Hortonworks HDP (<http://hortonworks.com/>)
- MapR (<http://mapr.com/>)
- Microsoft HDInsight (cloud, <http://www.windowsazure.com/>)

HDInsight distribution differentiator

HDInsight is an enterprise-ready distribution of Hadoop that runs on Windows servers and on Azure HDInsight cloud service. It is 100 percent compatible with Apache Hadoop. HDInsight was developed in partnership with Hortonworks and Microsoft. Enterprises can now harness the power of Hadoop on Windows servers and Windows Azure cloud service.

The following are the key differentiators for HDInsight distribution:

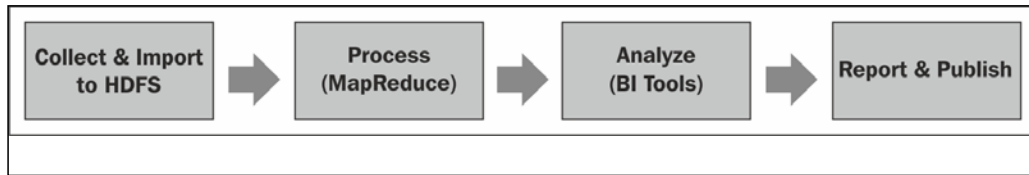
- Enterprise-ready Hadoop: HDInsight is backed by Microsoft support, and runs on standard Windows servers. IT teams can leverage Hadoop with **Platform as a Service (PaaS)** reducing the operations overhead.
- Analytics using Excel: With Excel integration, your business users can leverage data in Hadoop and analyze using PowerPivot.
- Integration with Active Directory: HDInsight makes Hadoop reliable and secure with its integration with Windows Active directory services.
- Integration with .NET and JavaScript: .NET developers can leverage the integration, and write map and reduce code using their familiar tools.
- Connectors to RDBMS: HDInsight has ODBC drivers to integrate with SQL Server and other relational databases.
- Scale using cloud offering: Azure HDInsight service enables customers to scale quickly as per the project needs and have seamless interface between HDFS and Azure storage vault.
- JavaScript console: It consists of easy-to-use JavaScript console for configuring, running, and post processing of Hadoop MapReduce jobs.

End-to-end solution using HDInsight

Since we are now familiar with the fundamentals, let's build an end-to-end Big Data solution using HDInsight.

Key phases of a Hadoop project

The following image shows the important phases of a typical project solution:



For any Hadoop project to get from data to information, we need the following phases:

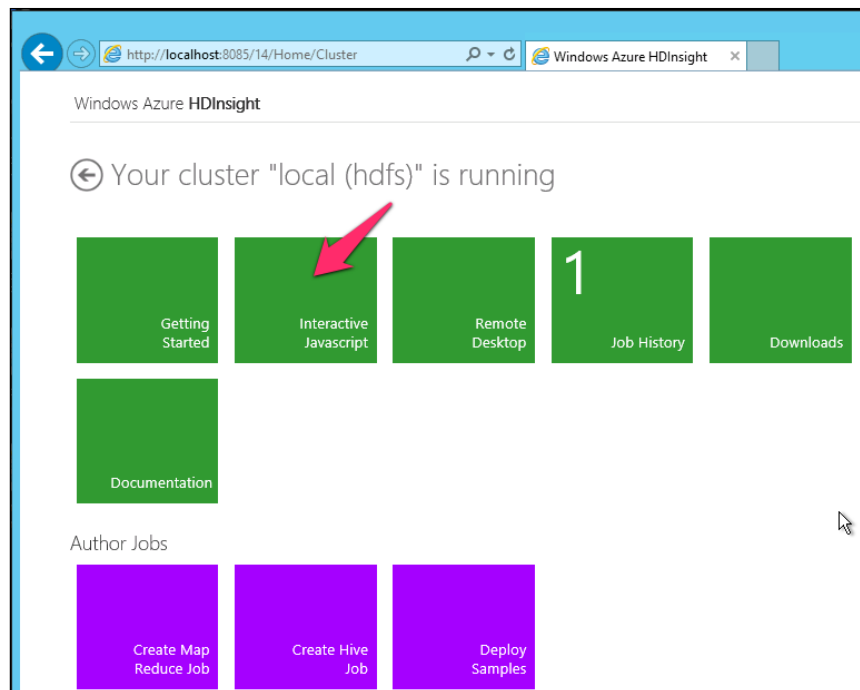
1. Collect data from external systems and ingest (load) it into HDFS.
2. Process data using MapReduce programs and save results back into HDFS.
3. Analyze results using Business Intelligence tools.
4. Publish reports for a wider audience.

Let's take an example that will count the number of occurrences for each word in a file. We will go from data collection to reporting, using HDInsight.

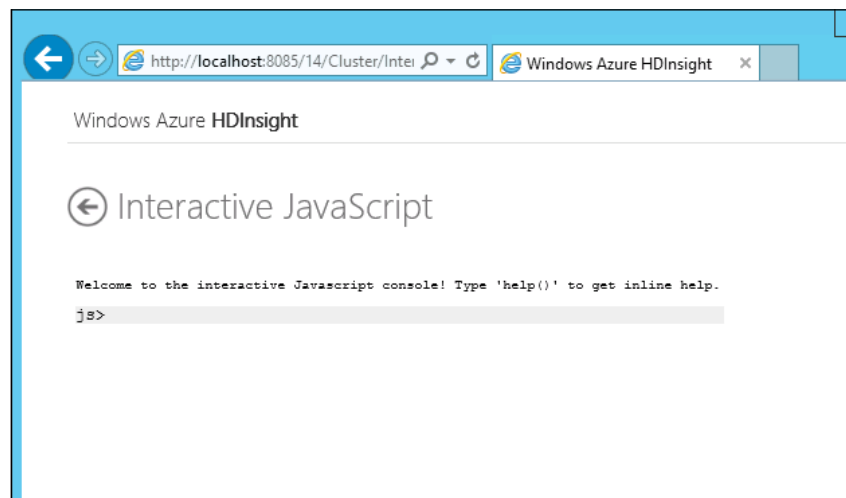
Stage 1 – collect data

For this example, we will use a NASA report on human spaceflight. The source of this example is http://www.nasa.gov/pdf/396117main_HSF_Cmte_FinalReport.pdf.

The text file is provided to you as `Nasa_Human_Spaceflight_Report.txt`. We will use the JavaScript console to upload this file to Hadoop. This feature is unique to HDInsight distribution, and makes it easy for development and troubleshooting. The following image shows the HDInsight cluster homepage:



After you click on **Interactive JavaScript**, you will see a console as shown in the following figure:



The following are the steps to load the file to HDFS:

1. First create a directory to keep all WordCount files in Hadoop using the JS console called `/user/hadoop/wcount`.
2. Next, upload the `Nasa_Human_Spaceflight_Report.txt` file.
3. Next, verify that the file is in HDFS using the `ls` command.

The following are the stage 1 commands for loading and verifying that the file is in HDFS:

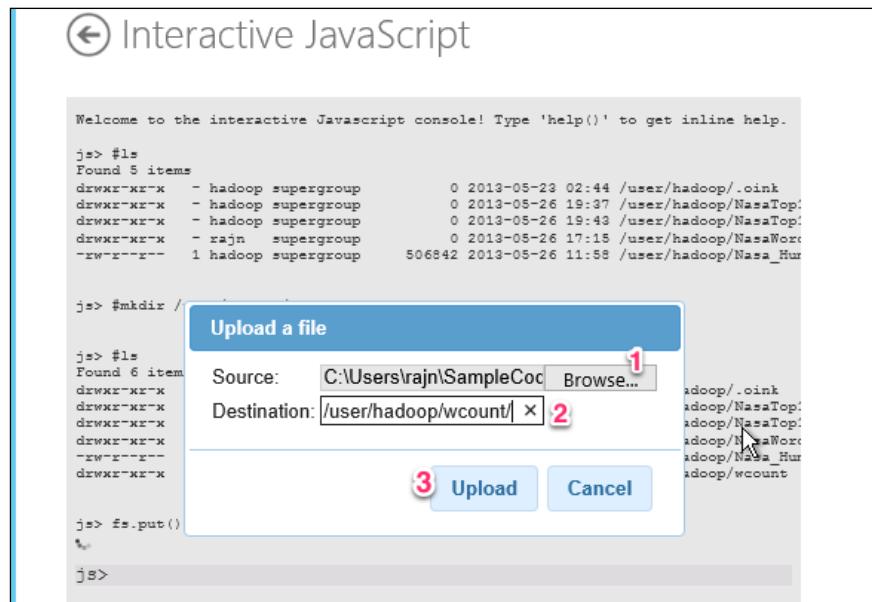
```
js> #mkdir /user/hadoop/wcount
```

```
js> fs.put()  
File uploaded.
```

```
js> #ls /user/hadoop/wcount  
Found 1 items  
-rw-r--r-- 1 hadoop supergroup 506842 2013-05-26 22:57 /user/  
hadoop/wcount/Nasa_Human_Spaceflight_Report.txt
```

```
js>
```

The following image shows JavaScript console File Upload:



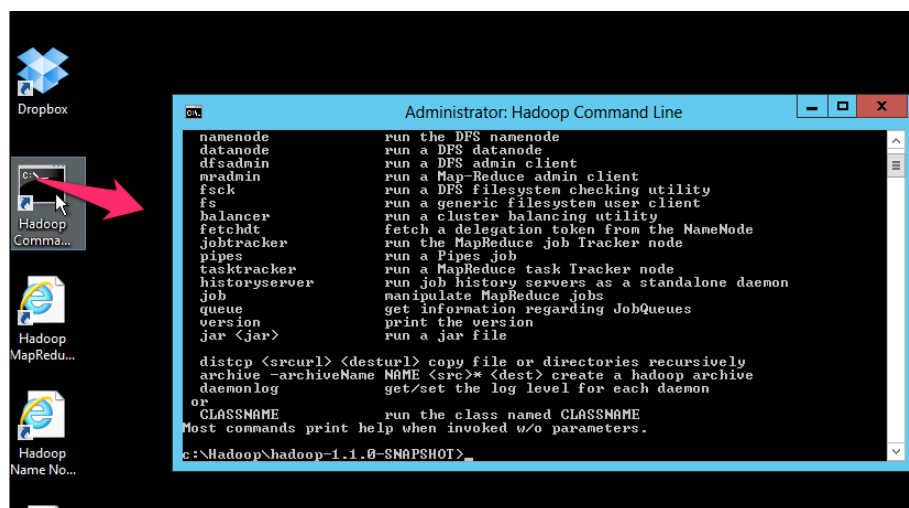
Stage 2a – process your data (build MapReduce)


To count the distinct words, you will need a MapReduce code. For this example, you can use the sample code provided in this book. There are three java files under the WordCount folder: WordCountMapper.java, WordCountReducer.java, and WordCount.java. The following are the steps to upload and compile the sample code:

1. First upload these files to your Hadoop client node. I have uploaded them to C:\Users\rajn\SampleCode\WordCount.
2. Next start the Hadoop command line (c:\hadoop\hadoop-1.1.0-SNAPSHOT\bin\hadoop.cmd).
3. Compile the java code to byte code using javac.
4. Then package the class files into a JAR file using jar.

Screenshots and code are shared. At the end, you should have the WordCount.jar file under C:\Users\rajn\SampleCode\WordCount.

The following image shows the Hadoop command line:



 The sample code here is in Java; the same code can be written in .NET, which integrates well with HDInsight.

The stage 2a commands will build the WordCount class files. They are as follows:

```
c:\>cd C:\Users\rajn\SampleCode\WordCount
```

```
C:\Users\rajn\SampleCode\WordCount>dir
```

```
05/26/2013  04:37 PM                954 WordCount.java
05/26/2013  04:38 PM                688 WordCountMapper.java
05/26/2013  04:39 PM                590 WordCountReducer.java
```

```
C:\Users\rajn\SampleCode\WordCount>c:\Hadoop\java\bin\javac -classpath
C:\Hadoop
```

```
\hadoop-1.1.0-SNAPSHOT\hadoop-core-*.jar *.java
```

```
C:\Users\rajn\SampleCode\WordCount>c:\Hadoop\java\bin\jar -cvf WordCount.
jar *.class
```

```
added manifest
```

```
adding: WordCount.class(in = 1412) (out= 810) (deflated 42%)
```

```
adding: WordCountMapper.class(in = 1691) (out= 738) (deflated 56%)
```

```
adding: WordCountReducer.class(in =5. 1704) (out= 719) (deflated 57%)
```

The following code is for WordCountMapper.java:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text,
IntWritable> {

    private Text word = new Text();
    private final static IntWritable ONE = new IntWritable(1);
```

```
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {

    // Break line into words for processing
    StringTokenizer wordList = new
        StringTokenizer(value.toString());

    while (wordList.hasMoreTokens()) {
        word.set(wordList.nextToken());
        context.write(word, ONE);
    }
}
```

The following code is for WordCountReducer.java:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable,
    Text, IntWritable> {

    private IntWritable totalWordCount = new IntWritable();

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

        int wordCount = 0;
        for (IntWritable val : values) {
            wordCount += val.get();
        }

        totalWordCount.set(wordCount);
        context.write(key, totalWordCount);
    }
}
```

The following code is for `WordCount.java`:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("Usage: WordCount <input dir> <output
dir>");
            System.exit(-1);
        }

        Job job = new Job();
        job.setJarByClass(WordCount.class);
        job.setJobName("WordCount");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setNumReduceTasks(1);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Downloading the example code



You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Stage 2b – process your data (execute MapReduce)

Now, we are ready to use the MapReduce code to get the frequency of words in the data file.

1. We will use the Hadoop command line to start MapReduce.
2. The initiating command is `hadoop jar WordCount.jar WordCount input output`.
3. Once the MapReduce job is completed, check if the output file is created in the location as specified. The output file that has the word frequency will have a name like `part-r-00000`.
4. You can view the contents of the file using `hadoop fs -cat outputfilename`.
5. Complete the stage 2b commands as shown in the following command:

```
C:\Users\rajn\SampleCode\WordCount>hadoop jar WordCount.jar
WordCount /user/hadoop/wcount/Nasa_Human_Spaceflight_Report.txt /
user/hadoop/wcount/NasaWordCountOutput
```

```
13/05/26 23:20:39 WARN mapred.JobClient: Use GenericOptionsParser
for parsing the arguments. Applications should implement Tool for
the same.
13/05/26 23:20:39 INFO input.FileInputFormat: Total input paths to
process : 1
13/05/26 23:21:15 INFO mapred.JobClient: map 100% reduce 100%
13/05/26 23:21:16 INFO mapred.JobClient: Job complete:
job_201305230235_0024
13/05/26 23:21:16 INFO mapred.JobClient: Map output
records=75119
```

```
C:\Users\rajn\SampleCode\WordCount>hadoop fs -ls /user/hadoop/
wcount/NasaWordCountOutput
Found 3 items
-rw-r--r-- 1 rajn supergroup 0 2013-05-26 23:21 /user/
hadoop/wcount/NasaWordCountOutput/_SUCCESS
drwxr-xr-x - rajn supergroup 0 2013-05-26 23:20 /user/
hadoop/wcount/NasaWordCountOutput/_logs
-rw-r--r-- 1 rajn supergroup 121653 2013-05-26 23:21 /user/
hadoop/wcount/NasaWordCountOutput/part-r-00000
```


Stage 3 – analyze data using JavaScript and Pig

Analyze your data. The previous step resulted in 10,622 unique words in the file (this is the reduced output result). If we have to find out the 20 most frequent words, we can change the MapReduce code we wrote in the second stage or leverage Pig, which is a wrapper on MapReduce. Let's see how this can be done using interactive JavaScript.

1. First, we call Pig to read the results file from stage 2.
2. The next step is to sort results by the count in descending order.
3. Then we take the first 20 records and store the results in `/user/hadoop/wcount/NasaTop20Words` directory.

The stage 3 commands are as follows:

```
js> pig.from("/user/hadoop/wcount/NasaWordCountOutput/part-r-00000", "word,count:long").orderBy("count DESC").take(20).to("/user/hadoop/wcount/NasaTop20Words")
2013-05-26 23:53:00,124 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success! (View Log)
```

```
js> file =fs.read("/user/hadoop/wcount/NasaTop20Words")
the      4819
of       2620
and      2092
...
```

```
js> top20words = parse(file.data, "word, count:long")
[
  0: {
    word: "the"
    count: 4819
  }
  ...
]
```

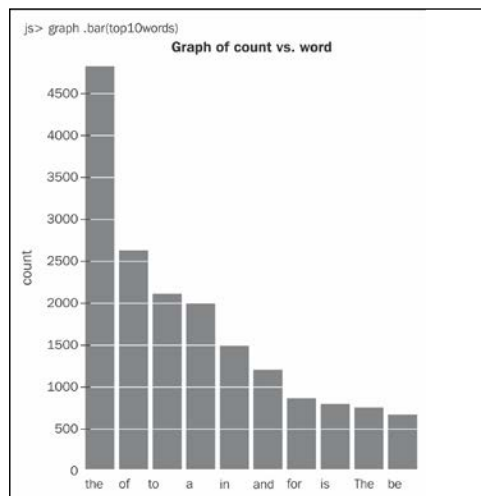
Stage 4 – report data using JavaScript charts

Data from the JavaScript console can be exported as a graph for reporting with the commands as shown as follows:

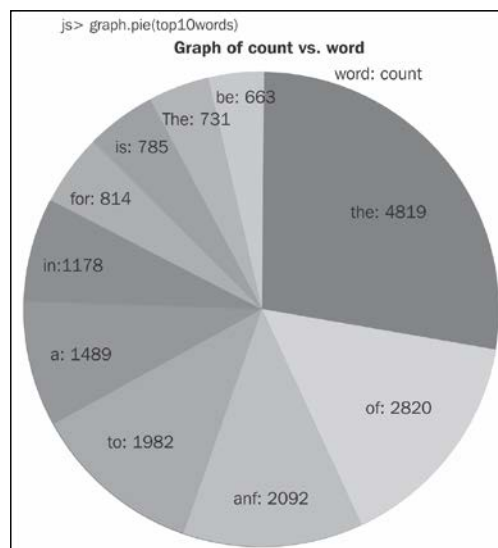
```
js> graph.bar(top20words)
```

```
js> graph.pie(top20words)
```

The following image shows the JavaScript bar chart:



The following image shows the JavaScript pie chart



Summary

In this chapter we started with today's challenging and complex Big Data needs. We then reviewed the Apache Hadoop components and the ecosystem of projects that provide a cost-effective way to deal with Big Data problems. We then looked at how Microsoft HDInsight makes the Apache Hadoop solution better by simplified management, integration, development, and reporting.

In the next chapter we will look at how to deploy a Microsoft HDInsight cluster on physical servers.

2

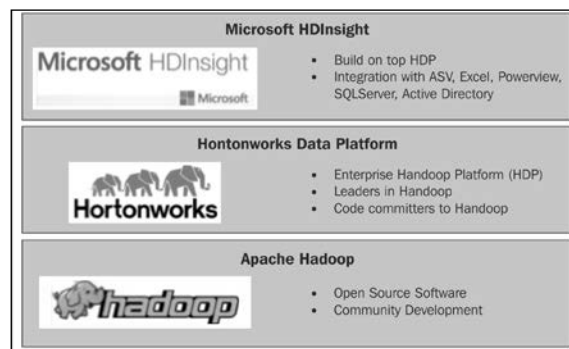
Deploying HDInsight on Premise

This chapter will articulate how to install and set up HDInsight on Windows servers. We will cover the following topics:

- HDInsight and Hadoop relationship
- Deployment options
- Single-node install
- Multinode planning and preparation
- Multinode install
- Managing HDInsight services
- Uninstalling HDInsight for Windows

HDInsight and Hadoop relationship

Before we get started with installation, it is good to understand what makes up HDInsight stack. The following screenshot shows the stack making up HDInsight:



The following are the components of HDInsight and Hadoop relationship:

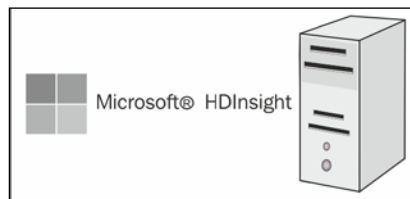
- **Apache Hadoop:** It is the open source software that allows distributed storage and computation. Hadoop is reliable and scalable.
- **Hortonworks Data Platform (HDP):** It is an enterprise-ready Hadoop platform. This is supported on Linux and Windows. It includes the following additional services:
 - **Operational services:** It serves to ease management of the Hadoop cluster
 - **Data services:** It serves to ease the storage, analysis, and access to Hadoop data
 - **Platform services:** It has high availability and snapshots on Hadoop
- **Microsoft HDInsight:** It is built with partnership of Hortonworks on top of HDP for Microsoft servers and Azure cloud service. It has the following additional key value added services provided by Microsoft:
 - Integration with Azure Storage Vault, Microsoft Active Directory, Excel, Powerview, SQL Server, .Net, C#, F#, and so on
 - JavaScript console for easy development

Deployment options for on-premise

HDInsight has two key modes of deployment: Windows Azure cloud service and Physical Windows Server. This chapter discusses the on-premise options.

Windows HDInsight server

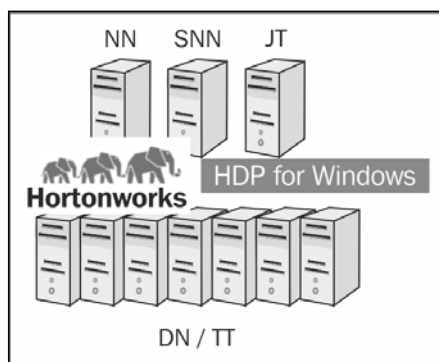
The following screenshot shows Microsoft HDInsight with a single node setup:



At the time this book was authored, Microsoft HDInsight on-premise would have been deployed only on a single node. This one-box solution can be used to test the functionality and development. You can promote your MapReduce code to a multinode Azure cloud infrastructure when you are ready.

Hortonworks Data Platform (HDP for Windows)

The following screenshot shows the Microsoft Hortonworks Data Platform cluster setup:



A single node setup is good for functionality development after which you will need a multinode cluster. HDP for Windows can be deployed on multiple Windows servers. With this option, you can scale the cluster as per your project needs.

Supported platforms for on-premise install

The following table lists the supported software platforms:

Type	Supported operating system
Single node	Windows 8, Windows 7, Vista SP2, XP SP3+, Server 2003
HDInsight server	SP2+, Server 2008, and Server 2008 R2
HDP for Windows	Windows 8, Windows 7, Server 2008 R2, and Server 2012

Single-node install

HDInsight single-node is a simple and excellent way for developers who want to try out MapReduce and HDFS features. You will need access to the Internet to complete this installation.

Downloading the software

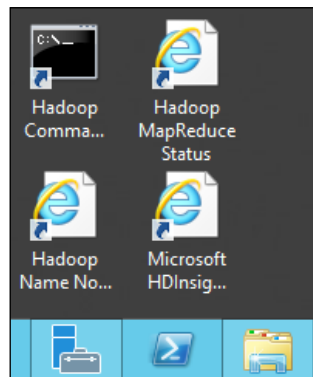
The link for the single-node install is <http://www.microsoft.com/web/gallery/install.aspx?appid=HDINSIGHT-PREVIEW>.

Running the install wizard

Double-click on the .exe file that you have downloaded, and then select the default options. The wizard will walk you through the following stages:

1. The prerequisites to install additional supporting software. Click on **I Accept**.
2. The install stage will download the necessary packages from the internet.
3. The configure stage will set up browser shortcuts on your desktop.

The following screenshot shows single-node post-install icons:

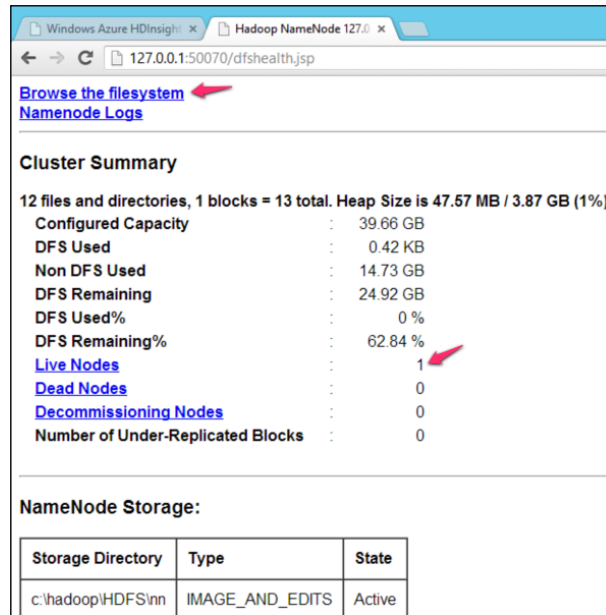


Validating the install

The following steps explain how to verify the install:

1. Click on the desktop icon **Hadoop NameNode**.
2. This should bring up the **HDFS Cluster Summary**.
3. Investigate the number of live nodes; it should be **1**.
4. You can then click on **Browse the filesystem**.

The following screenshot shows a valid single-node install:



Multinode planning and preparation

Following are the steps to prepare the nodes for a multinode cluster:


1. Install the following prerequisite software:
 1. Install Microsoft Visual C++ 2010 Redistributable Package from <http://www.microsoft.com/en-us/download/details.aspx?id=14632>.
 2. Install .Net Framework from <http://www.microsoft.com/en-us/download/confirmation.aspx?id=17851>.
 3. Install JDK 6 or higher. Download from the Oracle website. In the install directory, use a directory without any spaces, for example, C:\Java\jdk1.6.0_31. Do *not* use C:\Program Files\ as the install location. The link is <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u31-oth-JPR>.

4. Verify the install by starting the command prompt or Windows Powershell and type `java-version`.
 5. Add a system variable called `JAVA_HOME` and set it to the Java install directory, for example, `C:\Java\jdk1.6.0_31`.
 6. Open **Control Panel**. Click on **System** and then on **Advanced**.
 7. Under **System variables**, add **New**. Enter the variable name as `JAVA_HOME` and the directory as value.
2. Install Python as follows:
1. Download the .msi file from <http://www.python.org/download/>.
 2. Update the `PATH` environment variable and append the path where Python was installed, for example, `C:\Python33`.
 3. Navigate to **Control Panel | System**; Click on **Advanced**.
 4. Click on the **Environment variables** button. Edit `PATH` and add `C:\Python33` to the end of the values.
 5. Validate this by using Windows Powershell and type `python`, you will get a Python command prompt.

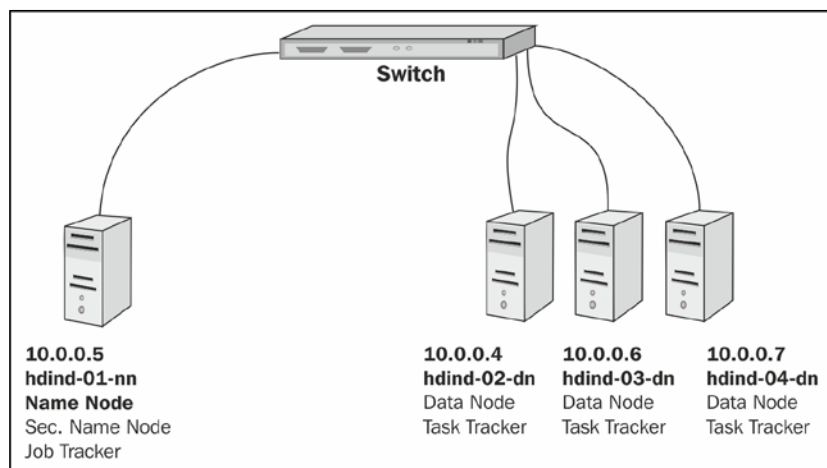
Setting up the network

The following steps are required to set up a network:

1. Each server must be able to connect to the other nodes in the cluster using IPV4 addressing. Disable IPV6 addressing for these nodes.
2. This can be done via a physical switch that all the nodes connect to, or a virtual network.
3. Once connected, each server should be able to ping any other node.

[ By default, ping is disabled on a Windows server. Use the following link to understand how to update the firewall to enable pings: [http://technet.microsoft.com/en-us/library/cc749323\(ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc749323(ws.10).aspx)]

4. You will need at least four nodes for a HDP cluster.
5. The following diagram shows a test bed that we will configure:



The following table lists the server and role:

Server IP	Name	Role
10.0.0.5	hdind-01-nn	NameNode + Secondary NameNode + JobTracker
10.0.0.4	hdind-02-dn	DataNode + TaskTracker
10.0.0.6	hdind-03-dn	DataNode + TaskTracker
10.0.0.7	hdind-04-dn	DataNode + TaskTracker

Setting common time on all nodes

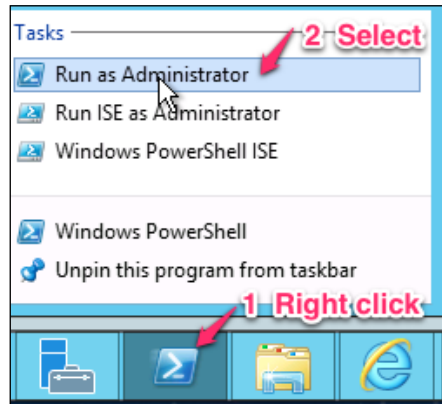
All nodes should have the exact date and time, which can be accomplished by configuring Windows time service. Microsoft has a support article that explains how to configure Windows server time service with a fix it wizard, which can be found at <http://support.microsoft.com/kb/816042>.

Setting up remote scripting

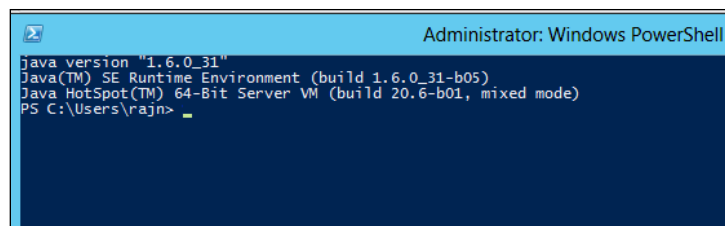
For a multinode environment setup, the installation will require Powershell scripts to be enabled on every node, and in addition run commands on other nodes using remote scripting. Follow the instructions to enable and verify remote scripting on every node:

1. Ensure that the administrator account on the Windows server node has a password. The remote scripting below will not work if the administrator account has an empty password.

2. Run Powershell as the administrator, as shown in the following screenshot:



The following screenshot shows **Administrator: Windows Powershell**:



1. In the command line, type the following to enable the trust between nodes:

```
C:\Users\rajn> Set-ExecutionPolicy "AllSigned"
```

```
C:\Users\rajn> Enable-PSRemoting
```

```
C:\Users\rajn> Set-item wsman:localhost\client\trustedhosts -value  
"hdind-01-nn.cloudapp.net,hdind-02-dn.cloudapp.net,hdind-03-dn.  
cloudapp.net,hdind-04-dn.cloudapp.net"
```

2. Use the following command to verify that the trust is established. Here, we will verify that hdind-01-nn can connect to hdind-02-dn, and run the hostname command on the remote server:

```
PS C:\Users\rajn> hostname  
hdind-01-nn
```

```
PS C:\Users\rajn> Invoke-Command -Computersname hdind-02-dn.  
cloudapp.net -ScriptBlock { hostname }
```

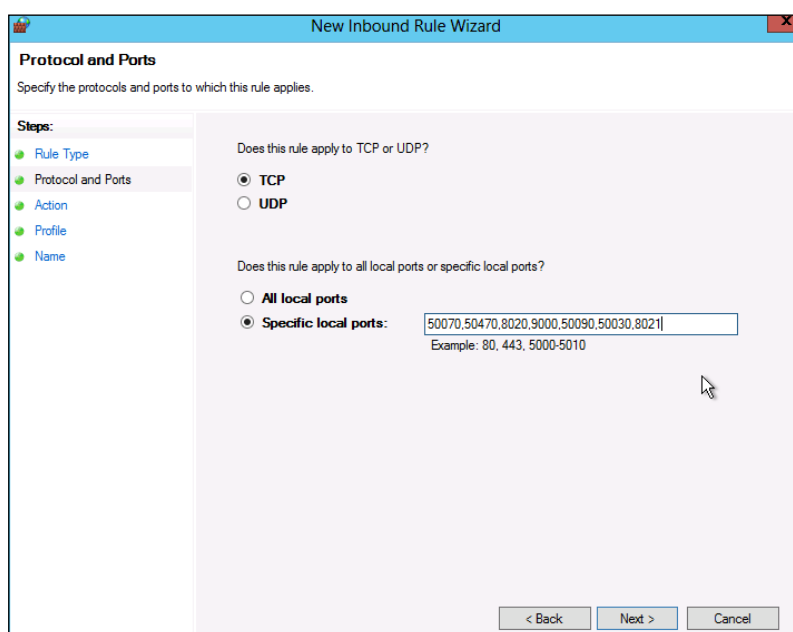
```
hdind-02-dn
```

```
PS C:\Users\rajn>
```

Configuring firewall ports

The following ports need to be opened for Hadoop to run on your machines. In single-node install, enable all the ports mentioned as follows. For HDP, depending on the role of the node, you will need to enable the respective ports in the following manner:

1. Navigate to **Start | Programs | Administrative tools | Windows Firewall**.
2. Click on **Inbound Rules**, and then on **New Rule**.
3. Select the protocol as **TCP** and specify the ports. Now click on **Next** and the **allow connection** option. Click on **Next**. Now, in the window that appears, click on **Next** again. Type a name, for example, **enable Hadoop ports**, and then click on **Finish** as, shown in the following screenshot:



The following table shows the server role and corresponding ports that need to be enabled:

Server role	Ports
NameNode	50070, 8020, and 9000
Secondary NameNode	50090
JobTracker	50030, 8021, 51111
DataNode and TaskTracker (slaves)	50075, 50475, 50010, 50020, and 50060
Hive server	10000 and 9083
Tempelton server	50111

Multinode installation

The following steps will guide you through the installation process.

Downloading the software

The Hortonworks website has the necessary software for HDP. Download this software for each node in the cluster. The link to the site is as follows:

<http://hortonworks.com/thankyou-hdp11-win/>

Configuring the multinode install

Create the `clusterproperties.txt` file based on the sample. Save it on your desktop. This defines your cluster.

```
#Log directory
HDP_LOG_DIR=c:\hadoop\logs

#Data directory
HDP_DATA_DIR=c:\hdp\data

#Hosts
NAMENODE_HOST=hdind-01-nn.cloudapp.net
SECONDARY_NAMENODE_HOST=hdind-01-nn.cloudapp.net
JOBTRACKER_HOST=hdind-01-nn.cloudapp.net
HIVE_SERVER_HOST=hdind-01-nn.cloudapp.net
```

```
OOZIE_SERVER_HOST=hdind-01-nn.cloudapp.net
TEMPLETON_HOST=hdind-01-nn.cloudapp.net
SLAVE_HOSTS=hdind-02-dn.cloudapp.net, hdind-03-dn.cloudapp.net, hdind-
04-dn.cloudapp.net

#Database host
DB_FLAVOR=derby
DB_HOSTNAME=hdind-01-nn.cloudapp.net

#Hive properties
HIVE_DB_NAME=hive
HIVE_DB_USERNAME=hive
HIVE_DB_PASSWORD=hive

#Oozie properties
OOZIE_DB_NAME=oozie
OOZIE_DB_USERNAME=oozie
OOZIE_DB_PASSWORD=oozie
```

Running the installer

On each node run the following command in the administration mode:

```
C:> msixexec /i "C:\Users\rajn\Downloads\hdp-1.1.0-GA\hdp-1.1.0-GA\
hdp-1.1.0-GA.winpkg.msi" /lv "C:\Users\rajn\Downloads\hdp-1.1.0-GA\
Install-Logs\HDP-1-1-install.log" HDP_LAYOUT="C:\Users\rajn\Desktop\
clusterproperties.txt" HDP_DIR="C:\hadoop" DESTROY_DATA="no"
```

Start the services by executing the following command on Powershell on the NameNode server:

```
cd %HADOOP_NODE_INSTALL_ROOT%
start_remote_hdp_services.cmd
```

Validating the install

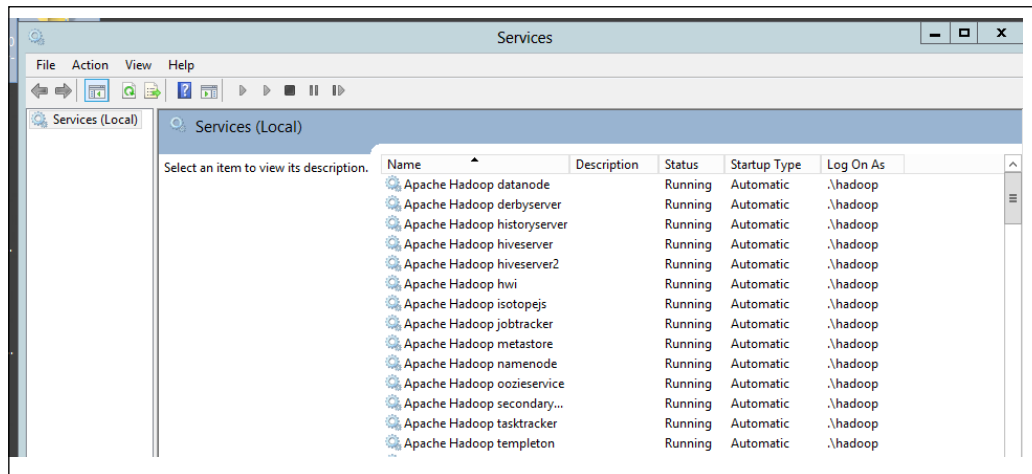
On the NameNode server (also JobTracker in our sample test bed), run the smoke test:

```
cd %HADOOP_NODE_INSTALL_ROOT%
Run-SmokeTests
```

This completes the multinode HDP for Windows setup.

Managing HDInsight services

Once HDInsight is configured, services are registered with the Windows system like other services, and can be stopped and started by navigating to **Control Panel | Administration | View local Services**, as shown in the following figure:



Uninstalling HDInsight

The steps for uninstalling HDInsight are as follows:

1. Launch the control panel. Next, click on **Programs**.
2. Select the program listed: **Hortonworks Data Platform 1.1 for Windows**.
3. With that program selected, click on the **Uninstall** option.



HDP for Windows cannot be uninstalled if there is data in existing data directories. First delete the non-empty data directories.

Summary

In this chapter we reviewed the Microsoft HDInsight stack and its relationship with Hortonworks Data Platform and Apache Hadoop. Next we went through the setup and installation process for a single-node and multinode cluster. A single-node setup is great for development and is quick to set up. Multinode setup needs planning and is set up at hardware, network, and software levels.

In the next chapter, we will see how to provision HDInsight service on the cloud using Windows Azure.

3

HDInsight Azure Cloud Service

This chapter will articulate how to install and set up HDInsight on Windows servers. The topics that we will cover are as follows:

- HDInsight Service on Azure
- Provision your cluster
- HDInsight Management portal
- Verify cluster and sample jobs
- Monitor your cluster
- Azure storage integration
- Remove your cluster

HDInsight Service on Azure

The following screenshot shows the Windows Azure HDInsight logo:



Windows Azure is a Windows cloud solution that allows you to rent, compute, and store resources on-demand for the duration of a project. HDInsight is a service that utilizes these elastic services and allows us to quickly create a Hadoop cluster for Big Data processing.

Considerations for Azure HDInsight Service

Azure HDInsight is a Platform as a Service that enables enterprises to leverage the power of Hadoop without the complexity and cost of installation, management, and support. HDInsight service has the following key advantages over the on-premise plan:

- You can start small and then grow as per your needs.
- You can provision the cluster in minutes without complex installation and setup.
- You get managed and highly available infrastructure from Microsoft support.
- You can shutdown your cluster when you don't need it. Optionally, you can retain data on Azure storage for future use.

Provision your cluster

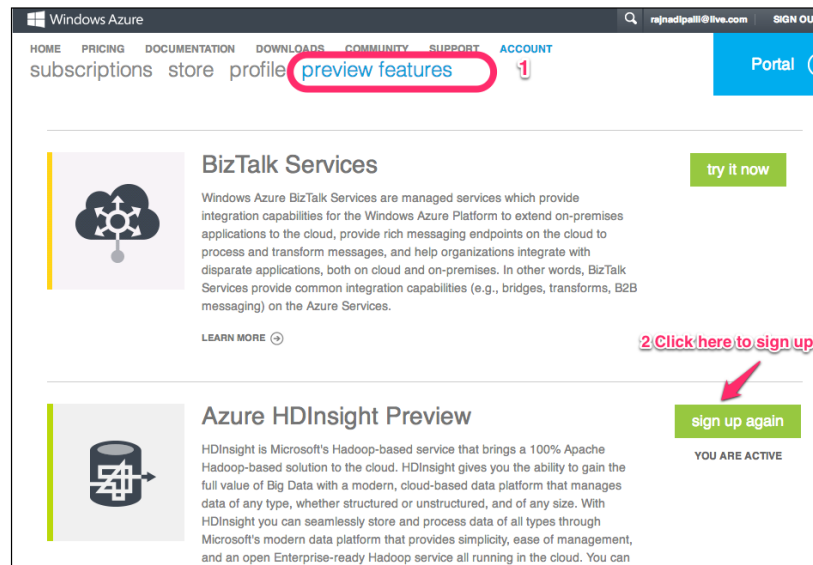
The following are the steps to provision your cluster:

1. Register for the Azure account.

First you will need an Azure account; register at this URL <http://www.windowsazure.com/en-us/> for a new account.

2. Request the preview feature.

The following is a screenshot on how to request a preview feature:



At the time this book was authored, HDInsight was a preview feature on Azure. A preview feature represents the latest capabilities added to the Microsoft Azure cloud platform. You will need to request access to this feature by going to Account | preview features, as shown in the previous figure. This request could take a few weeks to be approved.

3. Create a storage account.

Before you create a new HDInsight cluster, you will need a new storage account; this is where the HDFS data will reside. Follow these steps to create a new storage account:

1. Navigate to **NEW | DATA SERVICES**. Then, go to **STORAGE | QUICK CREATE**.
2. Enter a name and select a location. I have selected **North Europe**.
3. Do ensure **Geo-replication** is turned off.
4. Click on **CREATE STORAGE ACCOUNT** and you will have a storage account ready in a few minutes, as shown in the following screenshot:



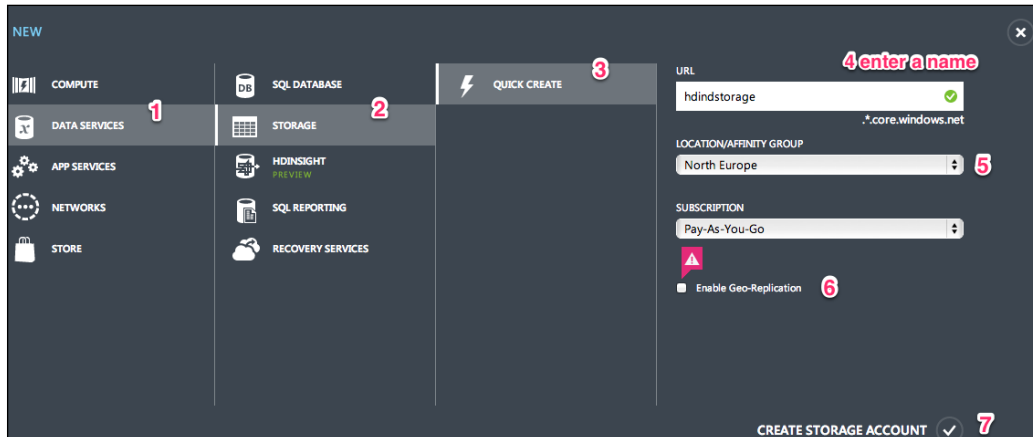
4. Create a HDInsight cluster.

Let's create a new HDInsight cluster as follows:

1. Navigate to **NEW | DATA SERVICES**. Click on **HDINSIGHT** and then **QUICK CREATE**.
2. Enter a name, select the size of the cluster, and then enter a password.

3. Select the storage account you created in the previous step.
4. Click on **CREATE HDINSIGHT CLUSTER**. This will provision the cluster and will take several minutes.

This eliminates the need for the complex software installation and setup we saw in *Chapter 2, Deploying HDInsight on Premise* on each node. The following screenshot shows the previous steps performed in the Azure site:



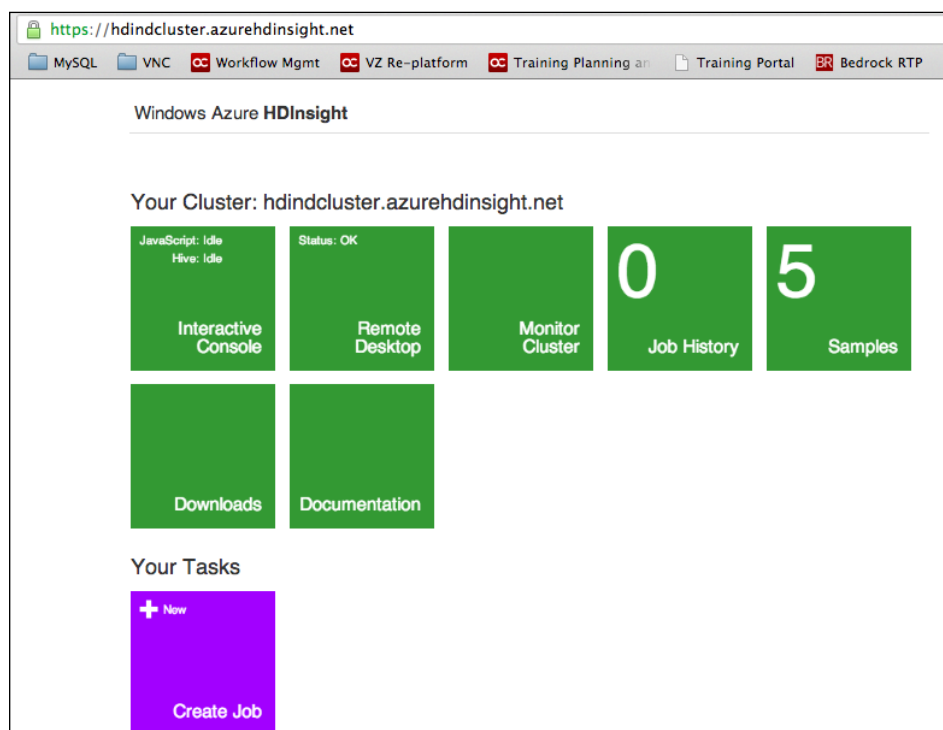
[At the time this book was authored, if you would have chosen the **CUSTOM CREATE** option, there would have been a single option available for storage location—**North Europe**.]

HDInsight management dashboard

Once you have created your HDInsight cluster, you can access your management portal using the link, as showing in the following screenshot:



Login with the username `admin` and the password you entered during the creation of the cluster. The following screenshot shows the management portal:



The managed dashboard can be used to monitor your cluster, see your job history, launch the JavaScript console, and create a new job.

Verify the cluster and run sample jobs

This section verifies the cluster that we just provisioned.

Access HDFS

To verify whether the distributed filesystem is accessible, we will use JavaScript. The following are the steps for verifying HDFS:

1. From the management dashboard page, click on **Interactive Console**. This will launch the JavaScript console, as shown in the following figure:

2. Explore the filesystem and verify one example file, as shown in the following code snippet:

```
js> #tail /example/data/gutenberg/davinci.txt
nd to
    let us know your plans and to work out the details.

WHAT IF YOU *WANT* TO SEND MONEY EVEN IF YOU DON'T HAVE TO?
Project Gutenberg is dedicated to increasing the number of
```



Deploy and execute the sample MapReduce job

To verify the MapReduce component, you can choose any of the sample jobs. I have selected the PI sample.

1. In the management portal, click on **Samples**, and then click on **PI estimator**.
2. Next, click on **Deploy to your cluster**.

- This sample has two parameters. The first one is the number of maps and I have selected 8. The second one is the number of samples and I have set that at 500,000. Next, click on **Execute job**.

This will execute the MapReduce job on the cluster, as shown in the following screenshot. The user interface ultimately executes the following Hadoop command:

```
c:\apps\dist\hadoop-1.1.0-SNAPSHOT\bin\hadoop.cmd jar c:\apps\Jobs\
templates\635066837930351814.hadoop-examples.jar pi 8 500000
```

Windows Azure HDInsight

← Create Job

Job Name and JAR File

Job Name:

JAR File: [Replace file](#)

☐ Delete existing JAR

Parameters

Parameter 0: [1 You can edit these](#)

Plain text

Final Command

[2 Click here](#)

Job History

Job Name	Command Line	Start time	End time	Status
No instances of this job have been submitted to this cluster.				

View job results

The output of the job is shown next using the standard out command `stdout`:

```
Number of Maps = 8
Samples per Map = 500000
...
Starting Job
Job Finished in 38.114 seconds
Estimated value of Pi is 3.1416040000000000000000
```

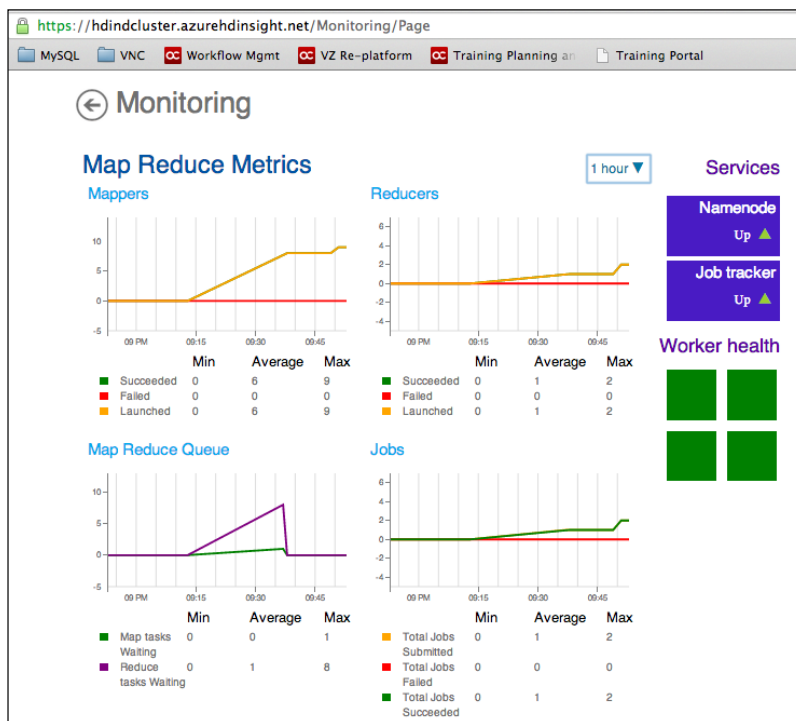

If your job has any errors, review the debug output (`stderr`) section. This will show both `INFO` and `ERROR` messages in Hadoop MapReduce. I have shown a portion of this output in the following code snippet:

```
13/06/13 01:36:42 INFO mapred.FileInputFormat: Total input paths
to process : 8
13/06/13 01:37:17 INFO mapred.JobClient: map 100% reduce 100%
13/06/13 01:37:19 INFO mapred.JobClient: Job complete:
```

Monitor your cluster

The dashboard provides you with a simple-to-use cluster monitoring. In the cluster management dashboard, click on **Monitor Cluster**. The following are the highlights of cluster monitoring:

- The dashboard shows the **NameNode** and **JobTracker** services on the top-right of the dashboard
- This is followed by four squares since we have four worker nodes in our cluster
- Additionally, the dashboard shows the trending charts for **Mappers**, **Reducers**, **Queue**, and **Jobs**, as shown in the following screenshot:

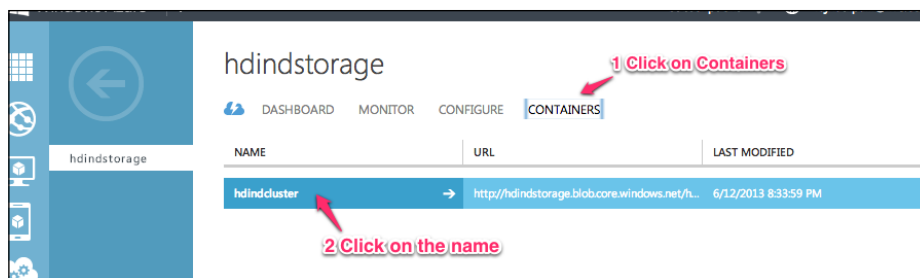


Azure storage integration

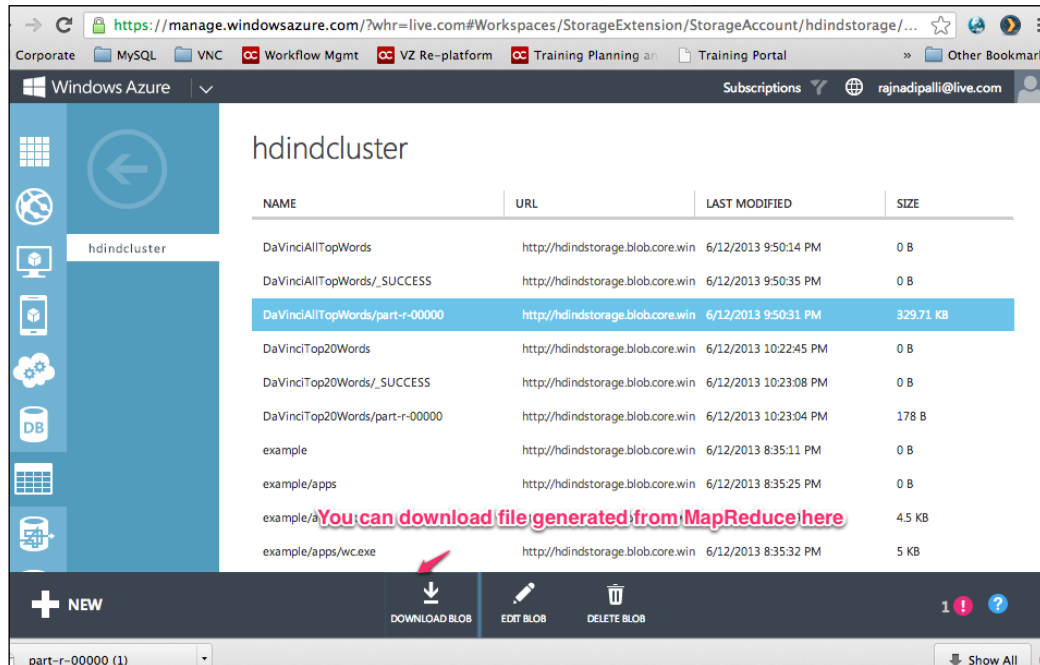
A key value-adding feature of HDInsight is its integration with Azure storage. This has two key benefits that I will highlight.

Firstly, the integration of HDInsight with Azure storage makes it easy for downloading results from MapReduce using the browser. You don't need to use Hadoop commands to get your MapReduce results. You can see the files listed on your screen without having to use the Hadoop commands, in the following manner:

1. Go to your Azure management portal at <https://manage.windowsazure.com>.
2. Navigate to **Storage | Select your Storage**.
3. Next click on **CONTAINERS**.
4. Click on the container you created for HDInsight; this will display a list of files also known as blobs.
5. You can select a file and then download it (link can be seen in the screenshot).
6. This allows you to browse HDFS files and download them without needing to use Hadoop `fs` commands, as shown in the following screenshot:



The following figure shows how to select a file and download it:



The second advantage of Azure storage integration is that you can retain your HDFS data even after you shutdown your HDInsight cluster. This allows you to pay only for long-term storage and use compute resources on-demand as needed.

Remove your cluster

The following section describes how to delete your cluster and restore it in case you need it later.

Delete your cluster

The first step to remove your cluster is to delete it using the Azure management page. Once you click on **Delete**, the request gets queued. Once you delete the cluster, you will not pay for any compute resources. This does not delete your storage, which enables you to revive the cluster if needed.

Delete your storage

If you do not need the data, delete the storage container first and then its components to completely clean your HDInsight cluster, in the following manner:

1. Go to your Azure management portal: <https://manage.windowsazure.com>.
2. Navigate to **Storage** | **Select your Storage**.
3. Next, click on **Container**. The **Delete** option is on the footer.
4. Next, go back to the storage listing. Click on **Storage** and then click on **Delete**.
5. This will now ensure your storage is completely deleted.

Restore your cluster

In case you preserved your storage and only deleted your cluster, follow the given steps to restore your cluster:

1. Go to your Azure management portal: <https://manage.windowsazure.com>.
2. Navigate to **HDInsight** | **Create new HDInsight cluster**.
3. Click on **CUSTOM CREATE**. Give a cluster name, select nodes, and give a password.
4. Next, click on **Use Existing Storage**. Select the account name and the container that has the HDFS data.
5. Click on **Next** and this will restore your HDInsight cluster, as shown in the following screenshot:

NEW

SQL DATABASE

STORAGE

HDINSIGHT

SQL REPORTING

RECOVERY SERVICES

QUICK CREATE

CUSTOM CREATE

CLUSTER NAME

*azurehdinsight.net

CLUSTER SIZE

4 data nodes

CLUSTER USER NAME ADMIN

PASSWORD

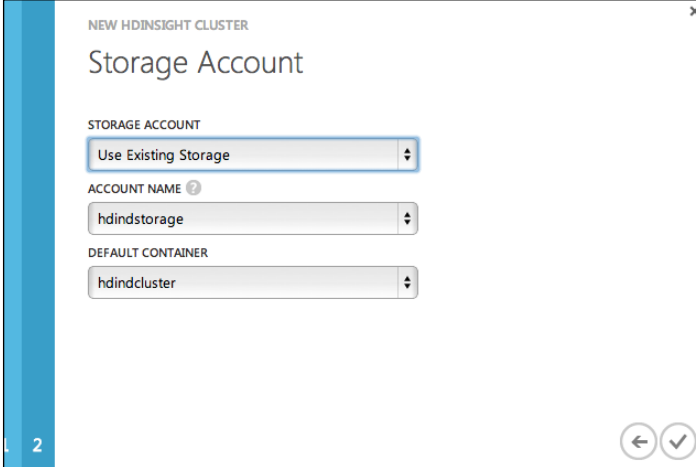
CONFIRM PASSWORD

STORAGE ACCOUNT

hdindstorage

CREATE HDINSIGHT CLUSTER

The following screenshot shows the final step to restore your cluster:



The screenshot shows a window titled "NEW HDINSIGHT CLUSTER" with a close button (X) in the top right corner. The main heading is "Storage Account". Below this, there are three dropdown menus:

- STORAGE ACCOUNT**: The dropdown menu is open, showing "Use Existing Storage" as the selected option.
- ACCOUNT NAME**: The dropdown menu shows "hdindstorage" as the selected option.
- DEFAULT CONTAINER**: The dropdown menu shows "hdindcluster" as the selected option.

At the bottom left of the window, there is a blue vertical bar with the number "2" inside it. At the bottom right, there are two circular buttons: a back arrow and a checkmark.

Summary

In this chapter, we reviewed how to register for an Azure account and provision a new HDInsight cluster in minutes. Azure HDInsight delivers Apache Hadoop services with a lower **total cost of ownership (TCO)** in comparison to an on-premise deployment by reducing the time to procure, set up, and manage the infrastructure. The additional benefit with the Azure storage vault is the ability to park your data at a low cost and not pay for the unused computer resources.

In the next chapter, we will cover common administration responsibilities including cluster health, NameNode, and JobTracker status.

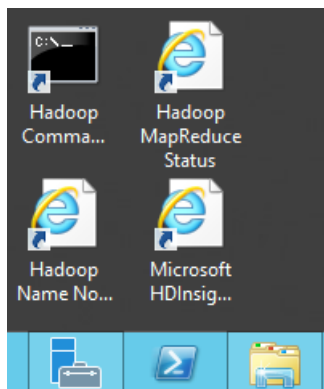
4

Administering Your HDInsight Cluster

This chapter will cover common administration responsibilities. We will cover the following topics:

- Cluster status
- Distributed filesystem health
- MapReduce health
- Key files and locations

The following screenshot shows the key shortcuts needed to administer your cluster:



Cluster status

After you set up HDInsight, there are three key icons on the desktop that help you monitor and manage the cluster. These are available on each node of the cluster, as shown in the following table:

Link name	URL	Purpose
Microsoft HDInsight Dashboard	http://namenode:8085	HDInsight Dashboard includes sample programs
Hadoop NameNode status	http://namenode:50070	Status for HDFS including NameNode and DataNodes
Hadoop MapReduce status	http://jobtracker:50030	JobTracker status for MapReduce

Distributed filesystem health

HDFS is a core component of Apache Hadoop, and as an administrator you need to watch out for how the cluster is serving your project needs. HDFS constitutes of one NameNode and several DataNodes. NameNode is the one in charge of this bookkeeping, while the actual data is hosted by the DataNodes.

NameNode URL

On your desktop, click on the **Hadoop NameNode Status browser** link and you will be directed to <http://namenode:50070>. This link allows us to access information of both the NameNodes and the DataNodes. The following are the highlights of what you can do from this page:

1. **Browse HDFS:** Use this to navigate to HDFS.
2. **Access the NameNode Log:** Use this to troubleshoot the NameNode issues.
3. **View the capacity, the percent used, and the percent available:** Use this information to plan for additional hardware that might be needed.
4. **Get the count of the nodes alive and dead with links for further information:** Use this information to diagnose nodes that need to be fixed as shown in the following screenshot:

NameNode '127.0.0.1:8020'

Started: Mon Jun 10 19:06:32 PDT 2013
 Version: 1.1.0-SNAPSHOT, r6e9bd1ba3514ee91bf4014e56980bf980f747b61
 Compiled: Fri May 17 21:57:23 Coordinated Universal Time 2013 by jenkins
 Upgrades: There are no upgrades in progress.

[Browse the filesystem](#) [NameNode Logs](#)

Cluster Summary

71 files and directories, 39 blocks = 110 total. Heap Size is 91.14 MB / 3.87 GB (2%)

Configured Capacity	: 39.66 GB
DFS Used	: 320 MB
Non DFS Used	: 15.54 GB
DFS Remaining	: 23.8 GB
DFS Used%	: 0.79 %
DFS Remaining%	: 60.02 %
Live Nodes	: 1
Dead Nodes	: 0
Decommissioning Nodes	: 0
Number of Under-Replicated Blocks	: 1

Browsing HDFS

The NameNode Summary page provides a link to browse HDFS. In the following screenshot, I have browsed to `/users/rajn/OTP` and the page shows a listing of four CSV files. The browsed page allows to download the file:

Contents of directory /user/rajn/OTP

Goto:

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
T_ONTIME_APR2013.csv	file	82.23 MB	1	64 MB	2013-06-10 20:00	rw-r--r--	Administrator	supergroup
T_ONTIME_FEB2013.csv	file	71.91 MB	1	64 MB	2013-06-10 20:00	rw-r--r--	Administrator	supergroup
T_ONTIME_JAN2013.csv	file	78.13 MB	1	64 MB	2013-06-10 20:00	rw-r--r--	Administrator	supergroup
T_ONTIME_MAR2013.csv	file	84.68 MB	1	64 MB	2013-06-10 20:00	rw-r--r--	Administrator	supergroup

[Go back to DFS home](#)

Local logs

MapReduce health

MapReduce is the second core component of Hadoop and handles all the processing of the data in HDFS. The JobTracker node is the Master node for MapReduce and tracks a job from start to finish. The actual worker bees are called **TaskTrackers**.

MapReduce summary

From your desktop, when you click on the Hadoop **MapReduce Status** browser link, you will be directed to the JobTracker URL at <http://jobtracker:50030>. In the next screenshot I have highlighted the key things you should watch for:

- Quick links at the top-right corner for **Running Jobs**, **Retired Jobs**, and **Local Logs**
- The summary table shows the number of nodes, the total number of slots and the ones that are occupied
- The **Filter** box will allow us to search for a specific job by its ID or the user ID
- **Running Jobs** shows in-progress jobs and the percentage of completion
- **Retired Jobs** shows execution details of completed jobs
- **Local Logs** allows us to get a detailed MapReduce log for each job
- Click on any job to get detailed information about it, as shown in the following screenshot:

The screenshot shows the Hadoop MapReduce Status browser interface. It includes a filter box at the top, a table of running jobs, a table of retired jobs, and a local logs section. Red arrows and text annotations highlight key features:

- 3 Search for a specific Job**: Points to the filter box.
- 4 Current Running Jobs and Progress**: Points to the 'Running Jobs' section header.
- 5 Execution History of past Jobs with Status**: Points to the 'Retired Jobs' section header.
- 6 Jobtracker Logs**: Points to the 'Local Logs' section header.

Running Jobs Table:

Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201306160611_0003	Tue Jun 18 17:12:21 PDT 2013	NORMAL	hadoop	TempletonControllerJob	0.00%	1	0	0.00%	0	0	1 running map tasks using 1 map slots. 0 additional slots reserved. 0 running reduce tasks using 0 reduce slots. 0 additional slots reserved.	NA

Retired Jobs Table:

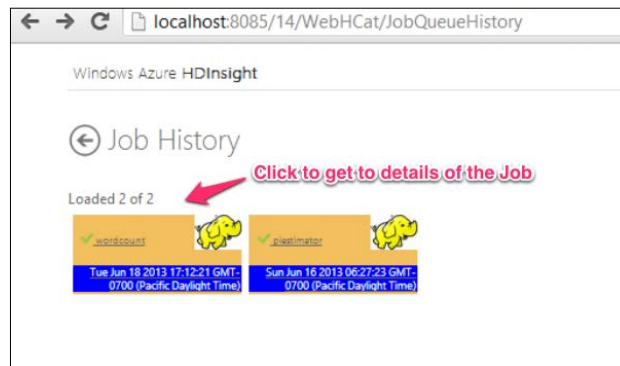
Jobid	Priority	User	Name	State	Start Time	Finish Time	Map % Complete	Reduce % Complete	Job Scheduling Information	Diagnostic Info
job_201306160611_0002	NORMAL	hadoop	PIEstimator	SUCCEEDED	Sun Jun 16 06:27:23 PDT 2013	Sun Jun 16 06:29:53 PDT 2013	100.00%	100.00%	0 running map tasks using 0 map slots. 0 additional slots reserved. 0 running reduce tasks using 0 reduce slots. 0 additional slots reserved.	NA
job_201306160611_0001	NORMAL	hadoop	TempletonControllerJob	SUCCEEDED	Sun Jun 16 06:26:44 PDT 2013	Sun Jun 16 06:30:11 PDT 2013	100.00%	100.00%	0 running map tasks using 0 map slots. 0 additional slots reserved. 0 running reduce tasks using 0 reduce slots. 0 additional slots reserved.	NA

Local Logs: Log directory, Job Tracker History

This is Apache Hadoop release 1.1.0-SNAPSHOT

MapReduce Job History

Another useful view is MapReduce Job History. This is reached via the HDInsight Dashboard page (click on **Job History** to access it). When you click on the tile, you will get details of that job, as shown in the following screenshot:



The page has two important shortcuts: **Output** and **Log**. **Output** allows you to revisit stdout as it happened at that time.



In Azure cloud version, HDInsight dashboard has additional useful features; specific to Job History it shows trending for Map and Reduce the steps.

Key files

The following are the key files in HDInsight and their locations. %hadoop% is your Hadoop home directory.

Filename	Node	Location
Core-site.xml	NameNode server	%hadoop%/conf
Hadoop-env.sh	NameNode server	%hadoop%/conf
Hdfs-site.xml	NameNode server	%hadoop%/conf
Mapred-site.xml	JobTracker server	%hadoop%/conf
Hadoop-core.jar	All nodes	%hadoop%/
Hadoop-client-1.1.0-SNAPSHOT.jar	All nodes and the client	%hadoop%/

Backing up NameNode content

NameNode is a single point of failure for Hadoop as it hosts critical metadata of HDFS.

The following command explains how to save the NameNode content to a local file, which can then be backed up onto an external storage if desired.

```
su hdfs
hadoop dfs -lsr / > hdfs-old-lsr-2013-06-21.log
```

The following command explains how to create a list of DataNodes available in the cluster:

```
su hdfs
hadoop dfsadmin -report > dfs-old-report-2013-06-21.log
```

The following command shows how to back up checkpoint files to a backup directory. Before copying, you need to save the namespace. Note that `dfs.name.dir` is defined in the `hdfs-site.xml` file:

```
hadoop dfsadmin -saveNamespace
dfs.name.dir/edits
dfs.name.dir/images/fsimage
```

Summary

Understanding how your cluster is being utilized is key to a successful Hadoop implementation. Hadoop provides several dashboards and metrics that report storage utilization, job performance, MapReduce slots, and trending. It is encouraged to periodically review your cluster usage as a team and then validate the results. This can be used to effectively utilize the cluster and/or plan additional hardware purchase.

In the next chapter we will discuss various methods to load data to your new cluster.

5

Ingesting Data to Your Cluster

Big Data Analysis first needs data to be collected and ingested to your cluster. This chapter will discuss various methods to load data to your new cluster.

We will cover the following topics:

- Loading data using Hadoop commands
- Loading data using Azure Storage Vault
- Loading data using Interactive JavaScript
- Shipping data to your Cluster
- Loading data from RDBMS via Sqoop

Loading data using Hadoop commands

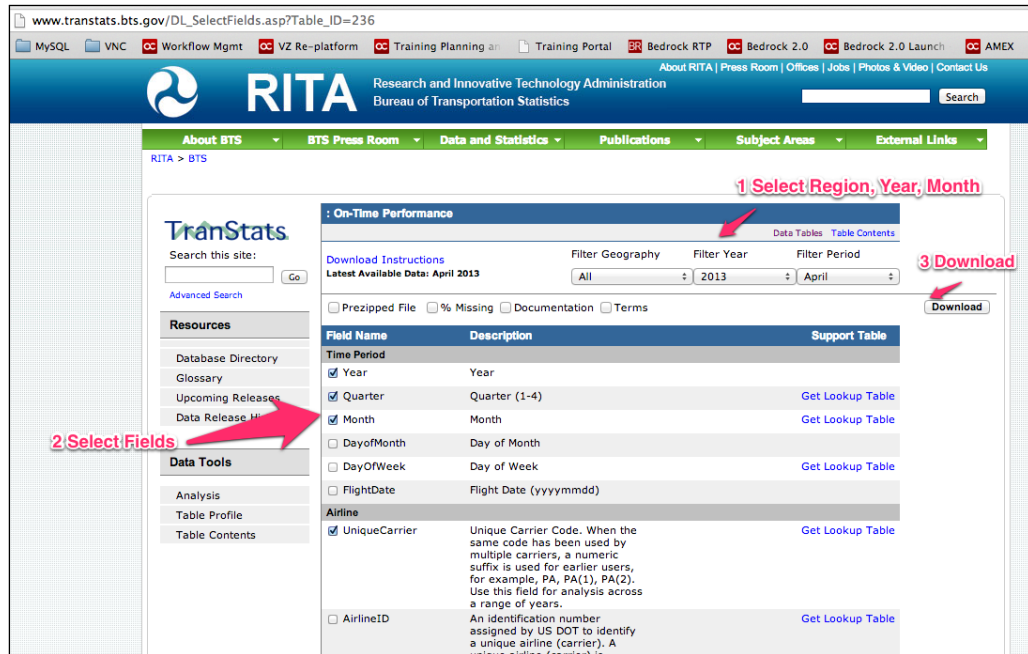
The simplest way to upload files is to use the Hadoop command line. The steps to load data are discussed in the next sections.

Step 1 – connect to a Hadoop client

You can choose any node of the cluster for development purposes. In a multi-developer environment, it is recommended that you have an edge node that acts as a Hadoop client, and all developers only access that instead of a Hadoop node.

Step 2 – get your files on local storage

Get your files on the edge node, either via Web download, SCP, or SFTP. As shown in the following screenshot, I am downloading the Airline On-time Performance data from the RITA website at http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236.



This downloads the files to your local directory and creates a ZIP file for each month. Unzip these files and arrange them in a single directory.



For each file, delete the first line, which has the column names. This will be of help in the next chapter where we transform the data.

Step 3 – upload to HDFS

Launch the Hadoop command line and use the following commands to upload the complete directory to HDFS:

```
C:\>hadoop fs -put C:\Users\Administrator\Downloads\OTP /user/rajn/
C:\>hadoop fs -ls /user/rajn/OTP
Found 16 items
-rw-r--r--    1 Administrator supergroup    55436563 2013-07-06 10:21 /
user/rajn/O
TP/T_ONTIME_APR2012.csv
...
-rw-r--r--    1 Administrator supergroup    56738357 2013-07-06 10:23 /
user/rajn/O
TP/T_ONTIME_OCT2012.csv
-rw-r--r--    1 Administrator supergroup    53836795 2013-07-06 10:23 /
user/rajn/O
TP/T_ONTIME_SEP2012.csv
```

Loading data using Azure Storage Vault (ASV)

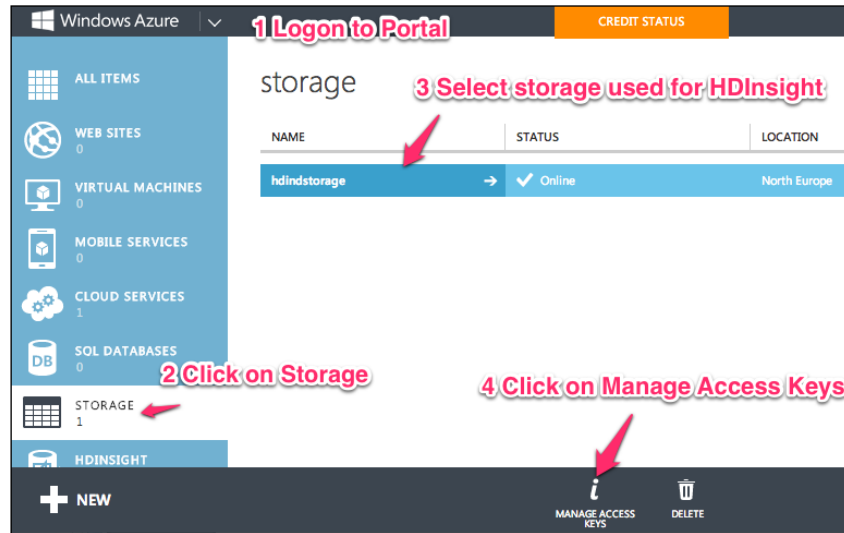
This section is applicable only if you are using the Azure cloud service. The HDInsight cluster on Azure uses Azure Storage.

Storage access keys

To use any of the ASV explorer tools, you will need your storage access keys. Perform the following steps to get this information:

1. Go to your Azure management Portal <https://manage.windowsazure.com>.
2. Click on **STORAGE** from your left-hand side menu.
3. Select the Storage used for the HDInsight cluster.
4. Click on **MANAGE ACCESS KEY** (bottom-center of the page as shown in the screenshot below). This brings up a pop up with account name, primary and secondary keys that you will need to connect to your storage remotely.

5. This will provide you a pop up with account name, primary, and secondary keys as shown in the following screenshot. You will need these details to connect remotely to your storage:



Storage tools

There are several tools that you can use to load data to Azure Storage; some of them are freeware. The following is a list of the popular ones, out of which we will learn about Azure Storage Explorer in detail:

- Azure Storage Explorer (<http://azurestorageexplorer.codeplex.com/>)
- Cloud Storage Studio (<http://www.cerebrata.com/products/cloud-storage-studio/introduction>)
- CloudXplorer (<http://clumsyleaf.com/products/cloudxplorer>)
- Windows Azure Explorer (<http://www.cloudberrylab.com/microsoft-azure-explorer-pro.aspx>)

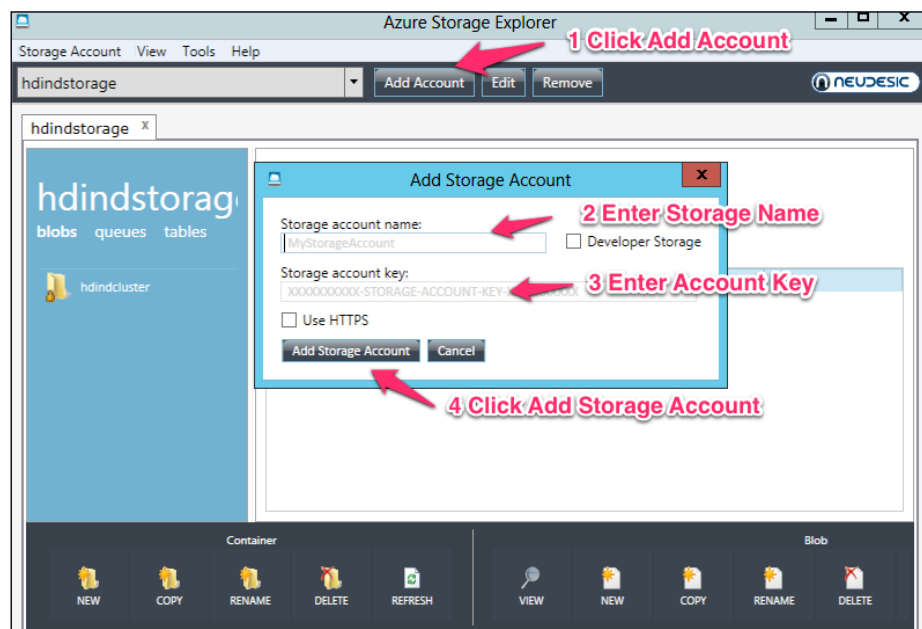
Azure Storage Explorer

Azure Storage Explorer is a GUI tool to view and edit Windows Azure Storage. This is available as a freeware under CCDL. First download and install the software from <http://azurestorageexplorer.codeplex.com/>.

Registering your storage account

Let's now register the storage used for HDInsight to Azure Storage Explorer by performing the following steps:

1. Click on the **Add Account** button.
2. Enter a storage name; I recommend using the same name as in Azure Portal.
3. Next, enter your Primary Key as obtained from Azure Portal.
4. Finally, click on **Add Storage Account** and now your storage will be added to the Explorer as shown in the following screenshot:

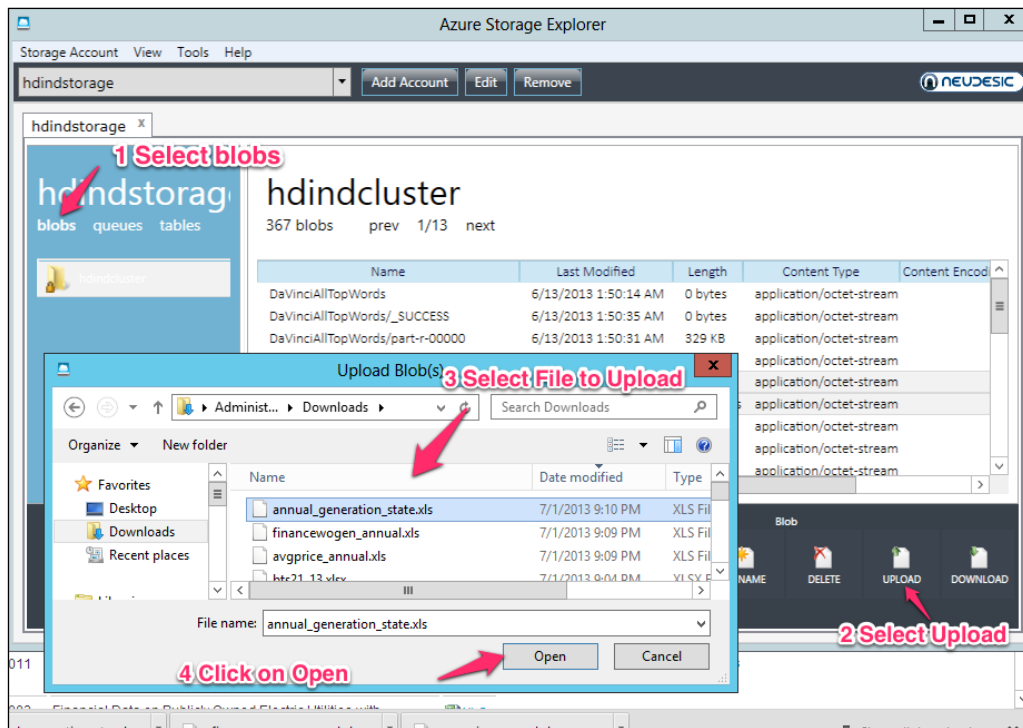


Uploading files to your blob storage

The following are the steps to upload files using Azure Storage Explorer with a screenshot:

1. Click on **blobs** on the left-hand side; you will see the right-hand side of the screen getting updated with all available blobs.
2. Click on the **UPLOAD** icon on the bottom-right corner of the screen.
3. Select the file(s) you want to be uploaded and then click on **OK**.
4. This will start the upload which you can verify by checking the Hadoop filesystem if needed.

5. You can use Azure Storage Explorer to download from Azure Storage to your local storage if needed. This option is available on the bottom-right corner:



Loading data using interactive JavaScript

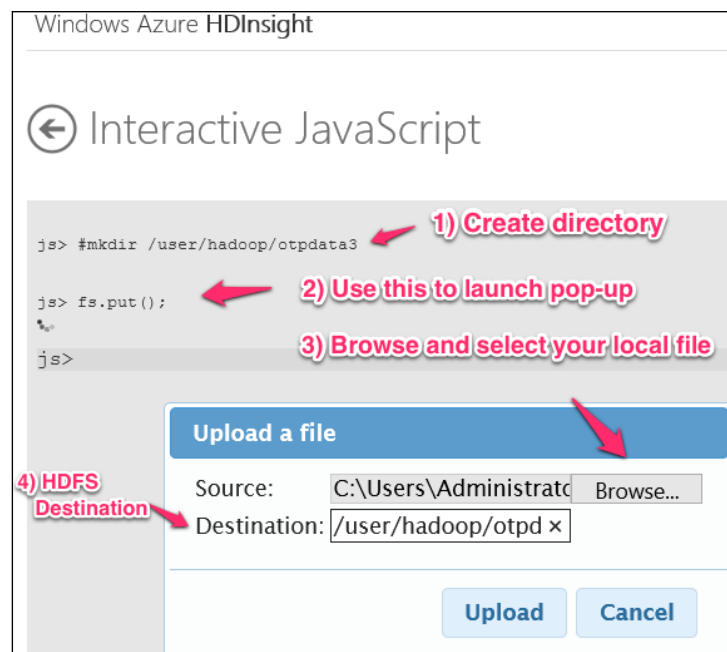
Another option to upload data to your cluster is to use the interactive JavaScript. This feature is unique to the HDInsight distribution. Perform the following steps to upload your files:

1. From your desktop, click on HDInsight Dashboard and then click on the **Interactive JavaScript** tile.
2. You will now see a JavaScript console.

3. Use the following commands on the JavaScript console:

```
js> #mkdir /user/hadoop/otpdata3
js> fs.put();
File uploaded.
js> #ls /user/hadoop/otpdata3
Found 1 items
-rw-r--r-- 1 hadoop supergroup 227977 2013-08-09 20:34 /user/
hadoop/otpdata3/Microsoft HDInsight Customer Previews.pdf
```

The following screenshot shows the steps for the JavaScript-based upload:



Shipping data to Azure

This section is applicable only if you are using the Azure cloud service. The HDInsight cluster on Azure uses Azure Storage.

Let's consider you want to analyze data from your **Enterprise Data Warehouse (EDW)**, which is 2 TB in size including the historical data. If you have a standard Internet connection, this would take about 100 days if you upload at a rate of 10 GB per day. A more practical solution for this is to download the data onto an external storage (Fireware or USB) from your local data center and then ship it to Microsoft Azure. They will load the data to your ASV. Contact Microsoft Azure Support for details on how to ship.

For ongoing incremental updates to your EDW that occur after you have shipped the storage, you can upload the deltas to keep the Hadoop data in sync with your EDW.

Loading data using Sqoop

Sqoop enables us to transfer data between any relational database and Hadoop.

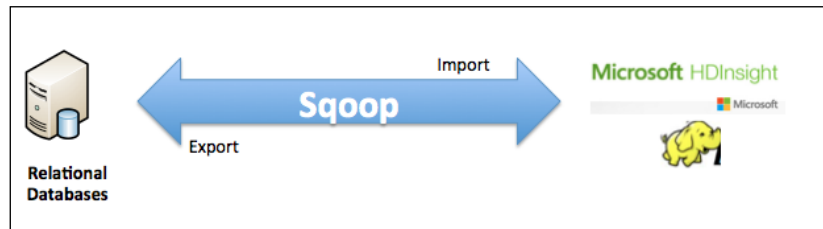
Key benefits

- Leverages RDBMS metadata to get the column datatypes
- Simple to script and uses SQL
- Uses MapReduce for export and import that enables parallel and efficient data movement

Two modes of using Sqoop

- **Sqoop Import:** Data moves from RDBMS to HDInsight
- **Sqoop Export:** Data moves from HDInsight to RDBMS

The following figure shows the two modes of using Sqoop:




Using Sqoop to import (SQL to Hadoop)

Let's consider the following scenario for import from Teradata database to Hadoop:

- **Source database:** Teradata
- **Server IP:** 10.11.12.6
- **SQL username:** dbc
- **SQL password:** dbc123
- **Table:** airline_otp (has 509,519 rows)
- **HDFS Directory:** /user/rajn/TDStage2

Following are the steps to import using Sqoop:

1. Ensure that you have the necessary JDBC drivers installed. Copy them to the Sqoop lib folder. In my installation, that was C:\Hadoop\sqoop-1.4.2\lib. I copied the terajdbc4.jar and tdgssconfig.jar files.


[ SQLServer drivers come preloaded with HDInsight.]

The following screenshot shows the location where the JAR files are copied to:



2. Use Windows PowerShell and run the Sqoop import command:


```
C:\Hadoop\sqoop-1.4.2\bin> .\sqoop import --connect
"jdbc:teradata://10.11.12.6" --username dbc --password dbc123
--table "airline_otp" --driver com.teradata.jdbc.TeraDriver
--target-dir /user/rajn/TDStage2 -m 1
```

[ -m 1 ensures that there is only one map task and hence only one file that has the complete dataset.]

3. Verify the data in HDFS by using a tail command:


```
C:\Hadoop\sqoop-1.4.2\lib>hadoop fs -tail /user/rajn/TDStage2/
part-m-00000
```

4. You can optionally filter the data by using a `where` clause. For the preceding dataset, let's filter for `flights` where the carrier is AA (American Airlines):

```
C:\Hadoop\sqoop-1.4.2\bin> .\sqoop import --connect
"jdbc:teradata://10.11.12.6" --username dbc --password dbc --table
"airline_otp" --driver com.teradata.jdbc.TeraDriver --target-dir
/user/rajn/TDStage3 --where " uniquecarrier='AA' " -m 1
```

Sqoop can also be used to export data out from Hadoop and send it to RDBMS. For further details, refer to the Sqoop user guide at http://sqoop.apache.org/docs/1.4.4/SqoopUserGuide.html#_literal_sqoop_export_literal.

Summary

In this chapter, we discussed the various options for moving data in/out of your HDInsight cluster. The Hadoop command line is good for scripting and it leverages the core Hadoop libraries. Azure Cloud Storage can be managed using several easy-to-use graphical tools. Sqoop is very handy for transferring data between relational databases and Hadoop.

In the next chapter, we will discuss how to transform/process the data you just ingested to the cluster using MapReduce, Hive, and Pig programs.

6

Transforming Data in Cluster

Once you have data in your cluster, you will need to process it. This chapter will discuss how to compute/transform data in Hadoop. We will cover the following topics:

- Transformation scenario
- MapReduce solution
- Hive solution
- Pig solution

Transformation scenario

In this chapter, we will look at solving a real-life problem statement—how to summarize data from different files in HDFS.

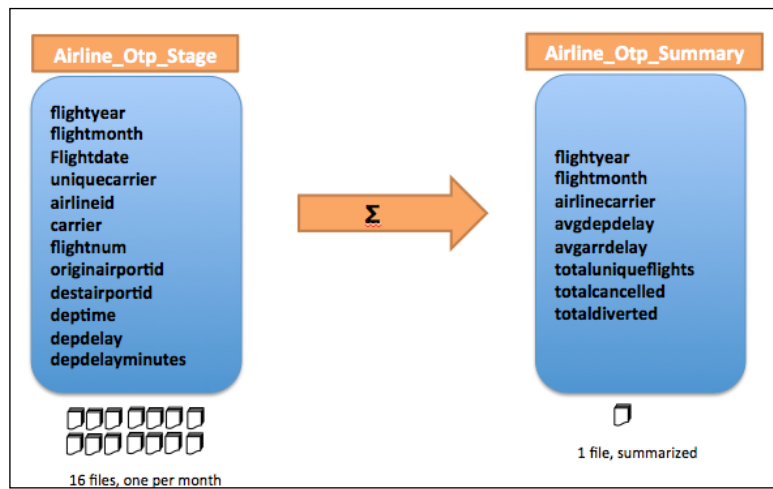
Scenario

Let's build a summary of the performance of an airline from the files loaded into HDFS. Data is downloaded from the RITA website at http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236 as discussed in *Chapter 5, Ingesting Data to your Cluster*.

Transformation objective

Summarize the data by year, month, and airline, and calculate the following:

- Average departure delay and average arrival delay
- Total unique flights, total cancellations, and total diversions
- The following figure shows the source and summary structures:



File organization

Before we start the transformation, we will organize the data files according to the year. Moving operations in Hadoop is very quick, as it just updates the references and does not move the blocks. Perform the follow steps to create two directories. There are 12 files in the 2012 directory, one for each month, and four files in the 2013 directory:

```
c:\> hadoop fs -mkdir /user/rajn/OTP/2012
c:\> hadoop fs -mkdir /user/rajn/OTP/2013
c:\> hadoop fs -mv /user/rajn/OTP/*2012*.csv /user/rajn/OTP/2012/
c:\> hadoop fs -mv /user/rajn/OTP/*2013*.csv /user/rajn/OTP/2013/
```

Now we will look into three different solutions commonly used with HDFS to solve this problem. MapReduce-based solutions offer maximum control, but require a lot of development. Pig and Hive are wrappers that work on top of MapReduce making it easier for non-developers, such as business analysts. But the drawback is that it allows less control on the actual code.

MapReduce solution

If you want to build your own application on top of HDFS, MapReduce provides low-level programming ability. MapReduce is also the foundation for other high-level projects such as Hive and Pig. Let's get into more detail of a MapReduce example with the airline on-time performance dataset.

Design

The approach for this problem statement is as described:

- We have a Map job that processes one line at a time (new line).
- The Map task emits a key, as a concatenated string of "Year + Month + Unique Carrier", and a value as a metric such as Departure Delay.
- The Reduce task receives all the values for a given key, for example, 2012, 01, AA, and does the math: *average departure delay = total departure delay / total number of entries*. The MapReduce will then run on all the files for a given year.

Map code

The process of calculating the average Departure Delay is shown in the following Java code. The same logic can be applied to other fields:

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;

/** key = year + month + Airline ID
 * value = depdelay
 * @author Raj Nadipalli */

public class OtpSumMapper extends Mapper<LongWritable, Text, Text,
DoubleWritable> {

    String otpDetails = null;
    @Override
```



```
public void map(LongWritable key, Text values, Context context)
throws
    IOException, InterruptedException {

    otpDetails = values.toString();
    //System.out.println("The line is " + otpDetails);
    String[] otpDetailsFieldValues = otpDetails.toString().
split(",");

    if(otpDetailsFieldValues.length == 19){

        // key is formed as year , month , carrier without the
double quotes ,
        String OtpKey =
otpDetailsFieldValues[0] + ',' + otpDetailsFieldValues[1] + ',' +
otpDetailsFieldValues[3].replace("\"","") + ',';

        // get the Departure Delay with is
11th string, so using [10]
        String val =
otpDetailsFieldValues[10];
        double dobval = 0;

        // if string length is 0, return 0, else convert to
double.
        if (val.trim().length() == 0) {
            System.out.println("Here in 0");
            dobval = -99999; // use this to indicate null, need to
exclude this for average
        }
        else {
            System.out.println("Here in NOT 0");
            dobval = Double.parseDouble(val);
        }

        DoubleWritable doubleval = new DoubleWritable(dobval);
        context.write(new Text(OtpKey), doubleval );
    }
}

}
```

Reduce code

The following is the Reduce Java code:

```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
/** key in  = year + month + Airline ID
 *  value in = arrdelay
 *  @author Raj */

public class OtpSumReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private DoubleWritable avgdepDelay = new DoubleWritable();
    @Override
    public void reduce(Text key, Iterable<DoubleWritable> values,
Context context)
        throws IOException, InterruptedException {

        double depDelay = 0;
        int otpcounter = 0;

        for (DoubleWritable val : values) {
            // -99999 is used to indicate null values
            if (val.get() != -99999) {
                depDelay += val.get();
                otpcounter += 1;
            }
        }
        // average is totdepDelay / count of records
        avgdepDelay.set(depDelay/otpcounter);
        context.write(key, avgdepDelay);
    }
}
```

Driver code

The following is the Driver code needed to run MapReduce:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class OtpSumDriver {

    public static void main(String[] args) throws Exception {

        if (args.length != 2) {
            System.out.println("Usage: OtpSumDriver <input dir> <output
dir>");
            System.exit(-1);
        }

        Job job = new Job();
        job.setJarByClass(OtpSumDriver.class);
        job.setJobName("OtpSummary");

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(OtpSumMapper.class);
        job.setReducerClass(OtpSumReducer.class);
        job.setNumReduceTasks(1);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Compiling and packaging the code

The following are the steps to compile and package the code:

1. Compile using `javac` with Hadoop core JAR file in the classpath.
2. Next, build the `OtpSummary.jar` file with all the classes:

```
c:\ch6code>c:\Hadoop\java\bin\javac -classpath C:\Hadoop\hadoop-1.1.0-SNAPSHOT\hadoop-core-*.jar *.java
c:\ch6code>c:\Hadoop\java\bin\jar -cvf OtpSummary.jar *.class
```

Executing MapReduce

Let's run the MapReduce code for the year 2012, which has 12 files per month:

```
c:\ch6code>hadoop jar OtpSummary.jar OtpSumDriver /user/rajn/OTP/2012 /
user/rajn/otpmapreducesum
13/07/14 05:42:44 WARN mapred.JobClient: Use GenericOptionsParser for
parsing the arguments. Applications should implement Tool for the same.
13/07/14 05:42:44 INFO input.FileInputFormat: Total input paths to
process : 12
13/07/14 05:42:44 INFO util.NativeCodeLoader: Loaded the native-
...
13/07/14 05:48:53 INFO mapred.JobClient:      Map output records=6114960
```

Results verification


The content of the `/user/rajn/otpmapreducesum/part-r-00000` file is as follows:

```
2012,1,AA,    4.958266527245265
2012,1,AS,    5.670788253477589
2012,1,B6,    6.874459603615743
2012,1,DL,    3.9864418469580367
2012,1,EV,    9.676645859489664
2012,1,F9,    9.2591671845867
2012,1,FL,    0.002310924369747899
2012,1,HA,    0.3807817033623485
2012,1,MQ,    4.238544624464117
2012,1,OO,    5.583140391496527
2012,1,UA,    7.905107187894073
2012,1,US,    2.82173209822085
```

Hive solution


As seen in the previous section, there is a lot involved to get a working MapReduce executable. I am a novice Java developer and it took me about four hours to get this working; for a non-developer, this would be a tougher task.

Hive enables business users who are typically aware of SQL and who are not MapReduce developers become more effective on using HDFS. Hive generates MapReduce code dynamically, based on the user's request. Hive comes preinstalled with HDInsight and HDP for Windows.

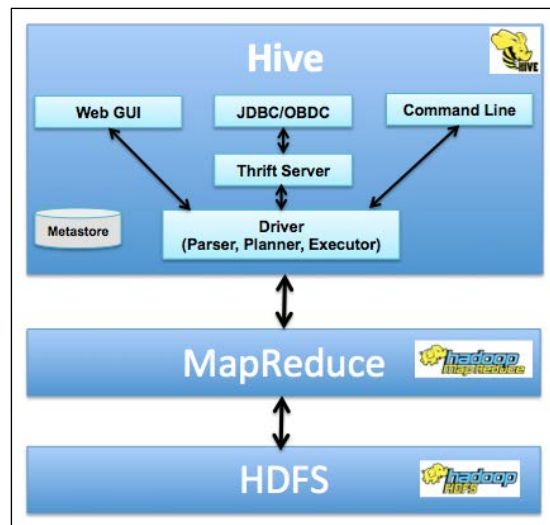
[ Hive syntax is very similar to MySQL.]

Overview of Hive

Apache Hive is a data warehouse infrastructure built on top of Hadoop and provides easy querying, summarization, and analysis. Its SQL-like interface is called **HiveQL**. Hive was initially developed by Facebook engineers.

[ Because Hive is based on Hadoop, it will not support update, delete, and row-level inserts.]

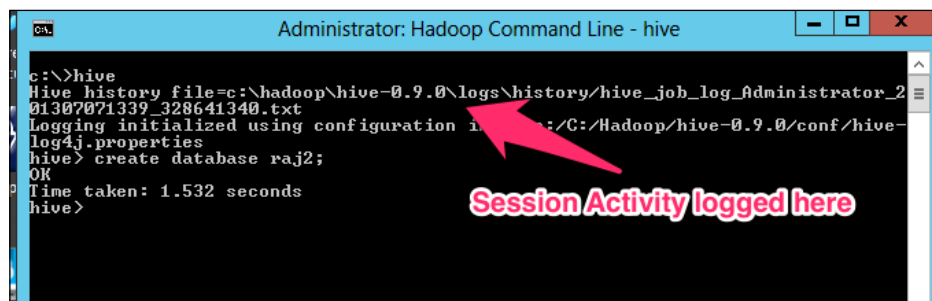
The following figure shows the Hive architecture and its relationship with Hadoop components:



- **Command line:** This is the best way to start using Hive.
- **Driver:** HiveQL is parsed and converted to a MapReduce Task using Driver.
- **HDFS:** MapReduce will access the necessary HDFS files.
- **Metastore:** This is typically a relational database and has the structure information of the files.
- **JDBC and ODBC drivers:** These drivers are supported and use the Thrift server to connect to Driver.

Starting Hive in the HDInsight node

To start Hive, click on the Hadoop Command Line icon on your desktop and then type `hive`. This will start the Hive session. It is recommended that you first create a separate database for each application. Here, I will be creating a database called `rajn`, as shown in the following screenshot:



```
c:\>hive
Hive history file=c:\hadoop\hive-0.9.0\logs\history\hive_job_log_Administrator_2
01307071339_328641340.txt
Logging initialized using configuration in file c:\Hadoop\hive-0.9.0\conf\hive-
log4j.properties
hive> create database raj2;
OK
Time taken: 1.532 seconds
hive>
```



The HDInsight service in Azure has a feature called **Interactive Hive** that simplifies access to Hive commands.

Step 1 – table creation

Next, we will create a table to store all the source files and call this `airline_otp_stage`. The table will be created as "external," which means that this Hive table is just a definition and will reference the HDFS files that we have already loaded earlier:

```
hive> use rajn;
hive> CREATE EXTERNAL TABLE airline_otp_stage (
    flightyear      SMALLINT,  -- COMMENT Year
    flightmonth     SMALLINT,  -- Month
    flightdate      STRING,    -- Flight Date (yyyymmdd)
    uniquecarrier   STRING,    -- Unique Carrier Code.
    airlineid       INT,       -- Identification number for carrier
    carrier         STRING,
    flightnum       STRING,    -- Flight Number
    originairportid INT,       -- Origin Airport,
    destairportid   INT,       -- Destination Airport, Airport ID
    depptime        STRING,    -- Actual Departure Time (hh:mm)
    depdelay        STRING,    -- Difference in minutes
    depdelayminutes STRING,    -- Difference in minutes
    depdel15        STRING,    -- (1=Yes) -> true/false
    arrtime         STRING,    -- Actual Arrival Time (hh:mm)
    arrdelay        STRING,    -- Difference in minutes
    arrdelayminutes STRING,    -- Difference in minutes
    arrdel15        STRING,    -- (1=Yes) -> true/false
    cancelled       STRING,    -- (1=Yes) -> true/false
    diverted        STRING     -- (1=Yes) -> true/false
)
partitioned by (flight_year INT)
row format DELIMITED
fields terminated by ',';
```

You can review if the table is created using the `describe` command:

```
hive> describe airline_otp_stage;
OK
```

Step 2 – table loading

Next, we will add partitions to the table, one each for 2012 and 2013:

```
ALTER TABLE airline_otp_stage ADD
PARTITION(flight_year = 2012)
LOCATION '/user/rajn/OTP/2012';
```

```
ALTER TABLE airline_otp_stage ADD
PARTITION(flight_year = 2013)
LOCATION '/user/rajn/OTP/2013';
```

```
hive> show partitions airline_otp_stage;
OK
flight_year=2012
flight_year=2013
```

Step 3 – summary table creation

Next, we will build the real summary table to store the results. This will fire a MapReduce job:

```
create table airline_otp_summary
as
select flightyear, flightmonth ,
regexp_replace(uniquecarrier,"\\",",") as airlinecarrier,
avg(depdelay) as avgdepdelay,
avg(arrdelay) as avgarrdelay, count(distinct flightnum) as
totaluniqueflights,
sum(cancelled) as totalcancelled,
sum(diverted) as totaldiverted
from airline_otp_stage
group by flightyear, flightmonth ,
regexp_replace(uniquecarrier,"\\",",");
```



```
Total MapReduce jobs = 1
Launching Job 1 out of 1
Starting Job = job_201307070550_0014, Tracking URL = http://
localhost:50030/jobdetails.jsp?jobid=job_201307070550_0014
Kill Command = c:\Hadoop\hadoop-1.1.0-SNAPSHOT\bin\hadoop.cmd job
-Dmapred.job.
tracker=localhost:50300 -kill job_201307070550_0014
Hadoop job information for Stage-1: number of mappers: 3; number of
reducers: 1
2013-07-07 14:11:35,743 Stage-1 map = 0%, reduce = 0%
```

Step 4 – verifying the summary table

Now, verify the summary table data:

```
hive> desc airline_otp_summary;
OK
flightyear      smallint
flightmonth     smallint
airlinecarrier  string
avgdepdelay     double
avgarrdelay     double
totaluniqueflights  bigint
totalcancelled  double
totaldiverted   double
Time taken: 0.235 seconds
hive> select * from airline_otp_summary limit 2;
OK
2012      1      AA      4.958266527245265      -0.27544268406337374
1254
718.0     93.0
2012      1      AS      5.670788253477589      3.081671849277483
390
402.0     58.0
```

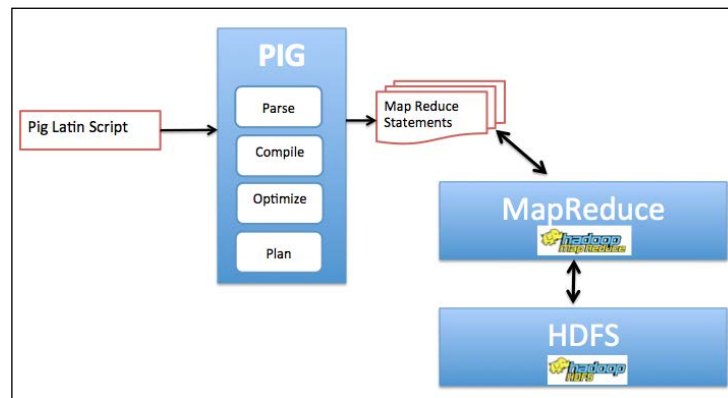
Pig solution

Pig is another high-level platform that generates MapReduce code dynamically. It is a scripting language similar to Python. The following are its key features:

- Rapid prototyping of algorithms
- Iterative processing of data (chaining)
- Joins are easy using Pig to correlate datasets
- Data can be verified onscreen or saved back to HDFS

Pig architecture

The following figure shows the Pig architecture:

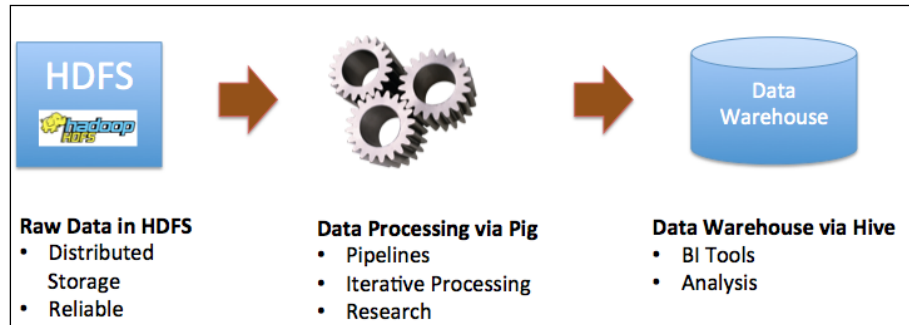


The preceding figure shows the following three steps:

1. Users start with a Pig script or the Pig command line (called **Grunt**).
2. Pig parses, compiles, optimizes, and fires MapReduce statements.
3. MapReduce accesses HDFS and returns the results.

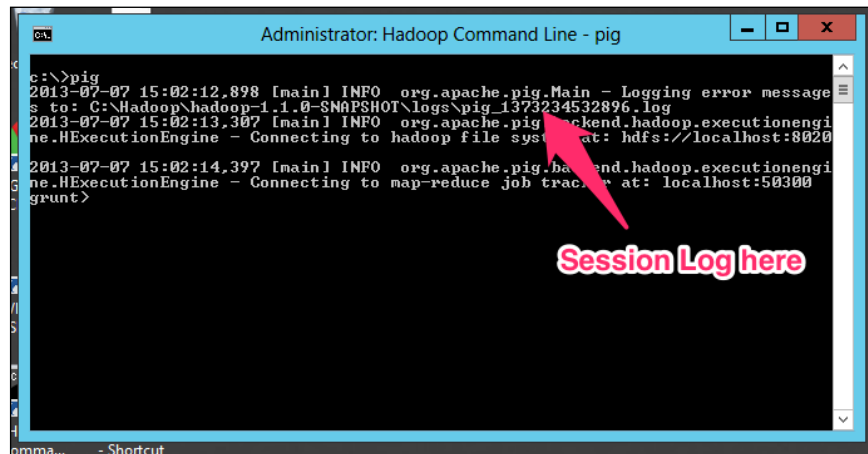
Pig or Hive?

If you are unsure whether to use Pig or Hive, think of a pipeline of ETL operations from **Raw Data in HDFS** to a BI analytical user, as shown in the following figure. Pig can be used for doing core ETL operations where the algorithms are known and Hive can be used for ad hoc queries:



Starting Pig in the HDInsight node

To start Pig, click on the Hadoop Command Line icon on your desktop and then type Pig, as shown in the following screenshot:



Pig Grunt script

The summarization can be achieved using five steps in Pig. I will first share the code here and then explain each line below.

Code

The following is a set of commands that you need to create a summary using Pig:

```
AirlineOTPStage = LOAD '/user/rajn/OTP' using PigStorage(',') AS
(flightyear:int, flightmonth:int, flightdate:chararray,
 uniquecarrier:chararray, airlineid:int, carrier:chararray,
 flightnum:chararray, originairportid:int, destairportid:int,
 deptime:chararray, depdelay:int, depdelayminutes:chararray,
 depdel15:chararray, arrtime:chararray, arrdelay:int,
 arrdelayminutes:chararray, arrdel15:chararray,
 cancelled:int, diverted:int);

AirlineOTPStageCleaned = foreach AirlineOTPStage generate flightyear,
flightmonth, REPLACE(uniquecarrier, '"', '') as newcarrier:chararray,
depdelay, arrdelay, cancelled, diverted;

GroupYrMoCa = group AirlineOTPStageCleaned by (flightyear,
flightmonth, newcarrier);

GroupYrMoCaCounts = foreach GroupYrMoCa generate FLATTEN(group)
as (Year, Month, Carrier), AVG(AirlineOTPStageCleaned.depdelay),
AVG(AirlineOTPStageCleaned.arrdelay), SUM(AirlineOTPStageCleaned.
cancelled), SUM(AirlineOTPStageCleaned.diverted);

STORE GroupYrMoCaCounts into '/user/rajn/AirlinePigSummary';
```

Code explanation

Let me explain the code shared in the previous section:

1. One neat feature of Pig is that it lets you define the schema on the fly. In the first line, `AirlineOTPStage` is defined as a variable that will hold all the 16 files under `/user/rajn/OTP` with a structure, that is, 19 comma-separated fields.
2. In the second line, `AirlineOTPStageCleaned` goes through every record of `AirlineOTPStage` and removes double quotes from the `uniquecarrier` column. It also reduces the number of columns to only those that we need for the next steps.
3. In the third line, `GroupYrMoCa` groups the `AirlineOTPStageCleaned` by `flightyear`, `flightmonth`, and `uniquecarrier`.
4. In the fourth line, `GroupYrMoCaCounts` computes the average and the sum of the columns.
5. In the fifth line, `STORE` is used to save the content to `/user/rajn/AirlinePigSummary`.

Execution

When the script is executed, you will see an information message as follows:

```
2013-07-08 07:14:37,045 [main] INFO  org.apache.pig.backend.hadoop.
executionengi
ne.mapReduceLayer.MapReduceLauncher - 0% complete
2013-07-08 07:14:37,358 [Thread-7] INFO  org.apache.hadoop.mapreduce.lib.
input.F
ileInputFormat - Total input paths to process : 2
2013-07-08 07:14:37,372 [Thread-7] INFO  org.apache.pig.backend.hadoop.
execution
engine.util.MapRedUtil - Total input paths to process : 16
2013-07-08 07:14:37,415 [Thread-7] INFO  org.apache.hadoop.util.
NativeCodeLoader
- Loaded the native-hadoop library
2013-07-08 07:14:37,416 [Thread-7] WARN  org.apache.hadoop.io.compress.
snappy.Lo
adSnappy - Snappy native library not loaded
2013-07-08 07:14:37,455 [Thread-7] INFO  org.apache.pig.backend.hadoop.
execution
engine.util.MapRedUtil - Total input paths (combined) to process : 15
2013-07-08 07:14:38,343 [main] INFO  org.apache.pig.backend.hadoop.
executionengi
ne.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_201307080428_0003
```

HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt
1.1.0-SNAPSHOT	0.9.3-SNAPSHOT	Administrator	2013-07-08 07:26:12	
2013-07-08 07:37:32	GROUP_BY			

Success!

Job Stats (time in seconds):

JobId	Maps	Reduces	MaxMapTime	MinMapTime	AvgMapTime
MaxReduc					
eTime	MinReduceTime	AvgReduceTime	Alias	Feature	Outputs
job_201307080721_0001	15	1	119	49	83
					536

```

536
536      AirlineOTPStage,AirlineOTPStageCleaned,GroupYrMoCa,GroupYrMoCaCou
nts
GROUP_BY,COMBINER      /user/rajn/AirlinePigSummary,

Input(s):
Successfully read 7989334 records (833188115 bytes) from: "/user/rajn/
OTP"
Output(s):
Successfully stored 225 records (12228 bytes) in: "/user/rajn/
AirlinePigSummary"
Counters:
Total records written : 225
Total bytes written : 12228
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 48
Total records proactively spilled: 437214
Job DAG:
job_201307080721_0001
    2013-07-08 07:37:32,933 [main] INFO
    org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Success!

```

Verification

Let's verify the content of the summarized file. I will first get the local file to my desktop and then view the content using Notepad:

```

c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -get /user/rajn/
AirlinePigSummary/part* C:\Users\Administrator\Desktop\

```

View the file contents...

```

2012  1  AA  4.958266527245265  -0.27544268406337374  718  93
2012  1  AS  5.670788253477589  3.081671849277483  402  58
2012  1  B6  6.874459603615743  1.9808709350737033  48  35
2012  1  DL  3.9864418469580367  -2.318164456812353  344  96
2012  1  EV  9.676645859489664  4.822557751676662  1074  149

```

Summary

To conclude this chapter, we have solved the airline summarization problem statement using three different solutions: MapReduce, Hive, and Pig. Hive and Pig do require getting used to the syntax, but are far superior in terms of the time needed to develop and ease of use by business analysts. In the next chapter we will discuss how to report and analyze the summarized data through BI tools and Excel.

7

Analyzing and Reporting Your Data

After you have ingested and transformed data in Hadoop, your ultimate goal becomes analyzing and reporting. You can make effective business decisions that improve customer satisfaction, engineer productivity, and/or identify new market segments. This chapter will discuss analyzing and reporting of the data in Hadoop. We will cover the following topics:

- Analyzing and reporting using Excel
- Using Hive for ad hoc queries
- Interactive JS (Azure) for analysis and reporting
- Other business intelligence tools

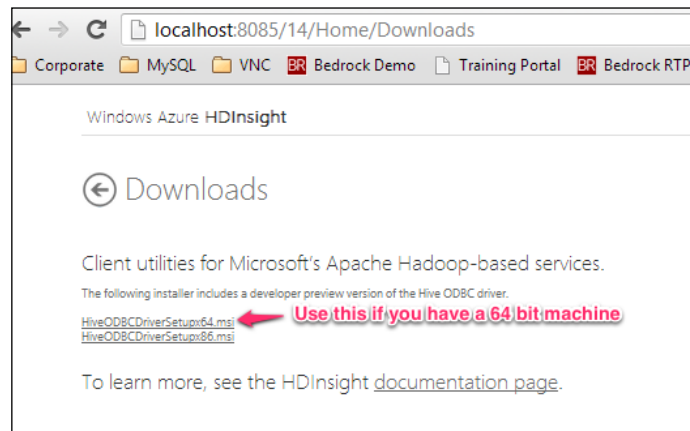
Analyzing and reporting using Excel

Excel is the most popular data analysis tool used by companies. HDInsight makes it easy to integrate Excel with Hadoop using Hive. To use this feature, you must have Excel 2010 or Excel 2013 or higher. Following are the steps to get your data into Excel and analyze it:

Step 1 – installing the Hive ODBC driver

The first step is to download the Hive ODBC driver and set it up. There are the following two options to get this file:

- Download the Hive ODBC driver from Microsoft Download Center at <http://www.microsoft.com/en-us/download/confirmation.aspx?id=37134>.
- From your HDInsight Dashboard, click on the **Downloads** tile. The following screenshot shows the options you will see after this. If you have a 32-bit node, select the **HiveODBCDriverSetupx86.msi** link, else select the **HiveODBCDriverSetupx64.msi** link:



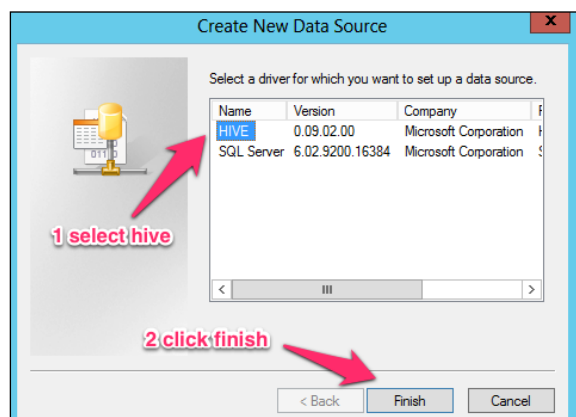
Once you find the driver MSI, double-click on it to install the driver. At the end of the installation, you will get a success message.

Step 2 – creating Hive ODBC data source

The next step is to configure Hive ODBC Data Source. Use the following steps to create your Hive ODBC Data Source:

1. Navigate to **Start | Control Panel**.
2. Then navigate to **System and Security | Administration tools**.
3. Next, click on **ODBC Data Sources** and select the user DSN.

4. Then, click on **Add**; this will enable you to add a new Data Source. You should now see an option with Hive as shown in the following screenshot. Click on **Finish**:



5. You will now see a **Hive Data Source Configuration** pop up. You will need to enter the following information into it:
 - **Data Source Name:** Provide a name.
 - **Description:** Provide a description.
 - **Host:** Use `localhost` if you have the single-box setup, or give a complete cluster name such as `hdindcluster.azurehdinsight.net` for the Azure HDInsight service.
 - **Port:** Provide the port number `1001` for single-box setup or type `563` for the Azure HDInsight service. This is the port of the thrift server.
 - **Authentication:** Select **No Authentication** for a single-box setup or provide the admin username and password for cluster in case of the Azure HDInsight service.

The following screenshot shows a new Hive Data Source configuration:

Hive Data Source Configuration

Data Source Name : rajnhive

Description : Local HDInsight Hive

Host : localhost

Port : 10001

Hive Server HTTP Path : /servlets/thrifts2

☐ Enable Logger ☐ Use Framed Packet Communication

Logging

Logger Severity : TRACE

Log File Directory : ...

Authentication

☒ No Authentication

☐ Username/Password

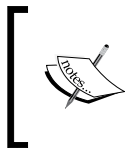
Username

Password (Will not be saved to configuration)

OK Cancel

Step 3 – importing data to Excel

Once you have the ODBC driver set up, you can use Excel 2010 or higher to analyze and visualize your data.

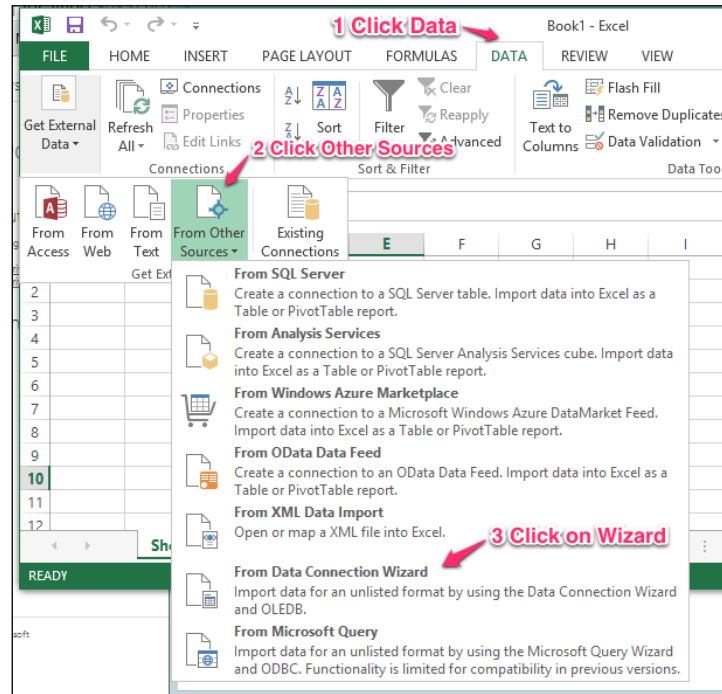


With Excel 2010, the PowerPivot tool is required for Excel in order to work on PivotChart/PivotTable. With Excel 2013, PowerPivot and PowerView are inbuilt; you just need to enable the COM add-ons: *Hive for Excel*, *Microsoft Office PowerPivot for Excel 2013*, and *PowerView*.

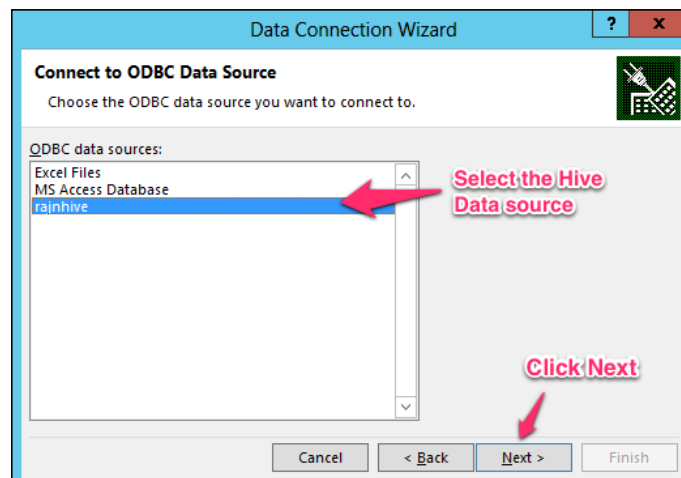
Perform the following steps to get your Hive data into Excel:

1. Open a new Excel workbook and click on the **Data** tab.
2. Click on **From Other Data Sources**.
3. Then click on **From Data Connection Wizard**.

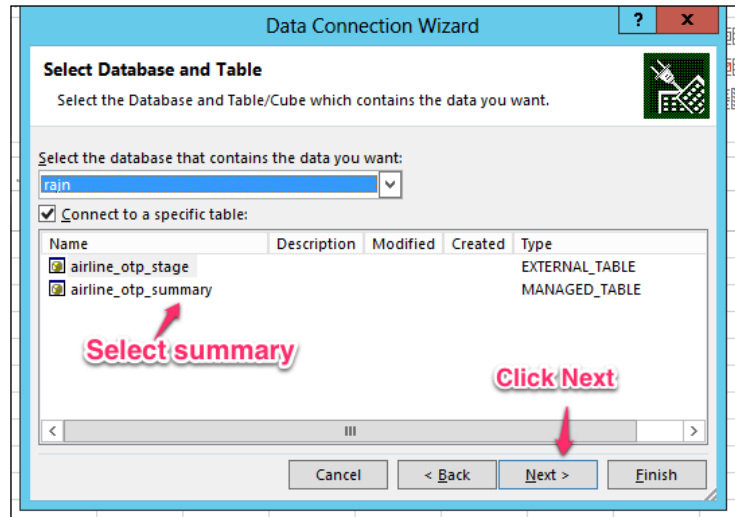
The following screenshot illustrates these steps:



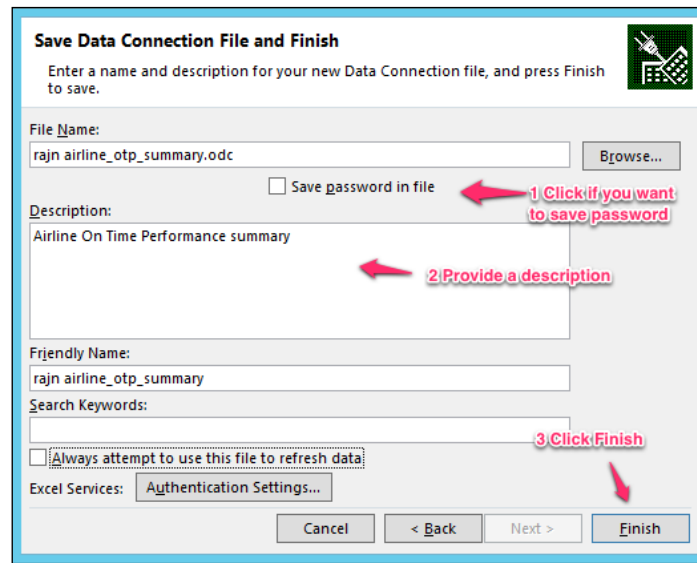
4. Next, the wizard will ask which remote data source you want to connect to; select **ODBC DSN**. The next pop up should show you Hive ODBC Data Source that you have just created. The following screenshot shows the driver I created:



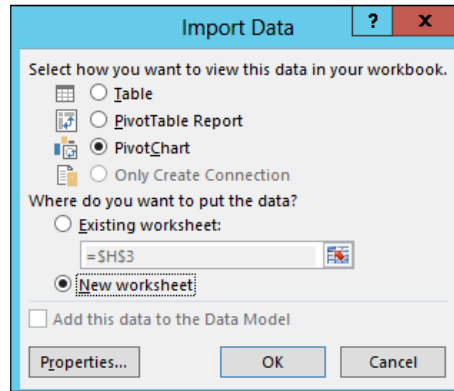
5. If you have a password set for the connection, you will be prompted again, otherwise, you will see the following screen that will allow you to select the table:



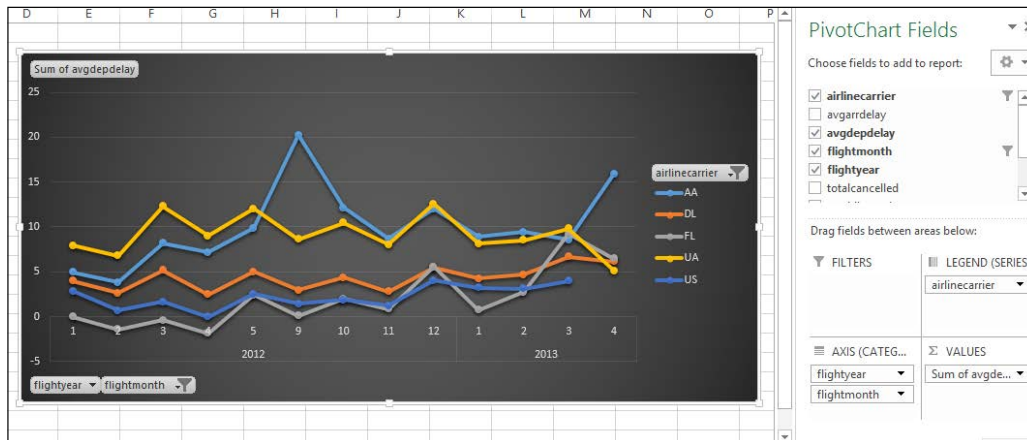
6. Next, you will get a pop up to save this Data Connection file. Add a description and click on **Finish**. The following screenshot shows this pop up:



7. Next, you have a choice to import the data as a simple table, PivotTable, or a PivotChart. Select **PivotChart**, as shown in the following screenshot:



8. Next, you will get options to configure the PivotTable and chart as shown in the following screenshot. Let's say you want to understand how the airlines have performed over time with regards to average Departure Delay, as shown in the following screenshot:



Let's understand how this works:

- Microsoft ODBC driver for Hive executes the HiveQL on the HDInsight cluster
- Hive will create a MapReduce job as per the query
- Results from the query are sent to the client PC where Excel is running
- All further analysis/charts will use data that is already imported to Excel

Hive for ad hoc queries

For business users who are aware of SQL and cannot wait for precomputed metrics, Hive is an excellent choice. It has a query language that converts SQL to MapReduce. The following section provides some examples. We will continue to use the airline on-time performance data as shown in *Chapter 5, Ingesting Data to your Cluster*.

Creating reference tables

In addition to the on-time performance stage data, we will load two additional lookup tables for a list of airports and a list of carriers. Data for this list can be found at the RITA website at http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236, and refer to the **Get Lookup Table** links.

```
c:\Users\Administrator\Desktop\Chapter 7>hadoop fs -put L_AIRPORT_ID.csv
/user/r
```

```
ajn/OTP/refdata/AIRPORT/L_AIRPORT_ID.csv
```

```
c:\Users\Administrator\Desktop\Chapter 7>hadoop fs -put L_UNIQUE_
CARRIERS.csv /u
```

```
ser/rajn/OTP/refdata/CARRIER/L_UNIQUE_CARRIERS.csv
```

```
-- Create managed Hive tables for Airport ID, Unique Carriers
```

```
create external table airport_list (
id INT,
description STRING,
state STRING,
longdescription STRING)
row format DELIMITED
fields terminated by ','
LOCATION '/user/rajn/OTP/refdata/AIRPORT';
```

```
create external table carrier_list (
carrierid STRING,
carrierdesc STRING)
row format DELIMITED
fields terminated by ','
LOCATION '/user/rajn/OTP/refdata/CARRIER';
```

Ad hoc queries

Now, your business users can run ad hoc queries to answer the following questions:

- What are the top 10 busiest airports?
- What is the percentage share of US Airways in the year 2012?
- Which carriers have most flights departing on time from NY state?
- Which carriers have improved their operations by reducing the departure and arrival delays from 2012 to 2013?

If you were to write a MapReduce program for each of these questions, your development team would be really busy and unable to meet business needs on time. This is where Hive becomes useful. I will share the solution of the first question in detail.

```
hive> select  description,  state,  count(flightnum) as flightcount
from    airline_otp_stage join airport_list on airline_otp_stage.
originairportid = airport_list.id
group by description,  state
sort  by flightcount DESC  limit 10;
```

The following is a screenshot of the Hive query execution:

```
MapReduce Total cumulative CPU time: 2 seconds 433 msec
Ended Job = job_201307201928_0008
MapReduce Jobs Launched:
Job 0: Map: 4 Reduce: 1 Cumulative CPU: 97.89 sec HDFS Read: 833446562 HDFS Write: 10881 SUCCESS
Job 1: Map: 1 Reduce: 1 Cumulative CPU: 2.308 sec HDFS Read: 11343 HDFS Write: 10881 SUCCESS
Job 2: Map: 1 Reduce: 1 Cumulative CPU: 2.34 sec HDFS Read: 11343 HDFS Write: 439 SUCCESS
Job 3: Map: 1 Reduce: 1 Cumulative CPU: 2.433 sec HDFS Read: 901 HDFS Write: 203 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 44 seconds 971 msec
OK
Atlanta GA 521356
Chicago IL 510365
Dallas/Fort Worth TX 365566
Houston TX 301883
Denver CO 300270
Los Angeles CA 286889
New York NY 263401
Phoenix AZ 230986
San Francisco CA 217441
Washington DC 190268
Time taken: 413.482 seconds
```


Analytic functions in HiveQL

Based on popular user requests, the Hive query language has been enhanced for windowing and analytical functions. These functions operate as per the SQL standards seen in other popular databases and are supported since release 0.11 (release dated May 2013). Following are the highlights:

- Windowing functions: Lead, Lag, first_value, and last_value
- The over clause: This aggregates with partition by and order by
- Analytics functions: Rank, Row_Number, Dense_Rank, Cume_Dist, Percent_Rank, and Ntile.

For further details, visit the Apache Hive wiki site at <https://cwiki.apache.org/Hive/languagemanual-windowingandanalytics.html>.

Interactive JavaScript for analysis and reporting

HDInsight comes with interactive JavaScript that interfaces well with Pig Latin. The following are the key benefits of the JavaScript console:

- Easy access to Hadoop cluster, such as for transferring files from the client to HDFS
- Interface to Pig Latin that enables a user to run Pig scripts
- Ability to display data as charts from the console

Refer to *Chapter 1, Hadoop & HDInsight in a heartbeat*, section *End-to-End solution using HDInsight*, Stages three and four.

Other business intelligence tools

As Hadoop's popularity is increasing, traditional BI companies are extending their support to this platform. I will provide a short list based on my familiarity; all of them require additional hardware and licensing:

- Microsoft SQL Server Analysis and Reporting Services (SSAS, SSRS): <http://www.microsoft.com/en-us/bi/default.aspx>
- SAS Enterprise Miner, SAS Visual Analytics: <http://www.sas.com/software/information-management/big-data/hadoop.html>

- Tableau software: <http://www.tableausoftware.com/solutions/hadoop-analysis>
- Platfora: <http://www.platfora.com/>
- Datameer: <http://www.datameer.com/>
- Oracle BI: <http://www.oracle.com/us/solutions/business-analytics/business-intelligence/overview/index.html>
- IBM Infosphere Data Explorer: <http://www-01.ibm.com/software/data/infosphere/data-explorer/>
- SAP BI Platform: <http://help.sap.com/bobip>

Summary

For any Hadoop project to be successful, the key is to gain actionable information from the vast amount of data collected in HDFS. Microsoft HDInsight enables analysis and reporting using Excel software on Big Data. Hive, Pig, and Interactive Java Script are other alternatives for ad hoc queries and analysis. With the increasing trend of Big Data, several traditional BI platforms including Microsoft now seamlessly integrate with Hadoop, enabling business users to model and visualize SQL and Big Data in one place.

In the next chapter we will review project planning and architectural considerations for your HDInsight-based project.

8

Project Planning Tips and Resources

HDInsight and Hadoop, in general, are relatively new platforms and require a new mindset from the decade's old relational and SQL systems. In this chapter, I will provide architectural and project considerations to help you on a successful HDInsight implementation. We will cover the following topics:

- Architectural considerations
- Project planning
- Reference sites and blogs

Architectural considerations

Although Hadoop is now a mainstream technology, and is being used in several public sectors and enterprises, the ecosystem is still evolving and new tools/projects are released every quarter. In the next few sections, I will highlight key architectural considerations.

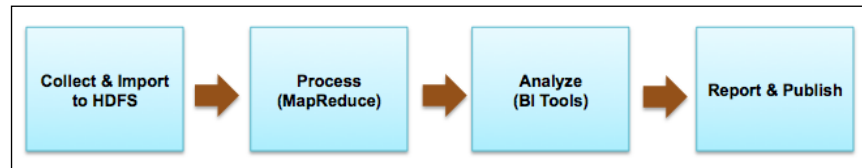
Extensible and modular

Every Hadoop project will go through the four steps as discussed in *Chapter 1, Hadoop and HDInsight in a Heartbeat*. The four steps are as follows:

1. Collect
2. Process
3. Analyze
4. Report

Use this model to design and build components for each module with well-defined interfaces. This allows you to have a pluggable model; for example, in future, if you find a better collection agent, you can replace the current one with little or no impact to your process or analyze layers.

The following figure shows the four-step model:



Metadata-driven solution

For each phase, consider a metadata-driven solution that will help in the development and streamline operations. The following are a few elements of the metadata repository:

- Data sources interfacing with the cluster; file types to be loaded into HDInsight with definitions, and expected frequency
- MapReduce workflows, priorities, scheduling, and notifications
- Downstream (target) systems to the HDInsight cluster

Integration strategy

Plan and build a good integration strategy for both upstream and downstream systems. Typical implementations involve an edge node that is dedicated to receive files and ingests to HDInsight. For sending data out of HDInsight, you can set up scheduled workflows to export data out of the cluster to the external system or have the downstream system query HDFS via Hive/Pig.

Security

Hadoop essentially inherits **POSIX** filesystem security with three roles: users, groups, others, along with read/write/execute permissions for each role. This allows basic filesystem security that can be used to manage access by functional users defined per application. Hadoop also integrates with Kerberos for network-based authentication.

If your data has **Personal Identifiable Information (PII)**, consider masking and/or tokenizing to ensure the information is protected.

Project planning

In this section, I will share a model that you can apply to your existing project or a new project on HDInsight.

Proof of Concept

The first phase of the project should be a time-bound evaluation with specific goals. The following should be in your **Proof of Concept (POC)** plan:

- **POC objectives:** The volume of data to be processed, desired aggregates, integration and query support, and success criteria.
- **Non-functional considerations:** Will this architecture scale? Will it be available? Can it meet the SLAs as data grows? Is it maintainable?
- **Timeline and resources:** Start and end dates with resources identified.
- **POC readout:** POC should have a closure date with a readout to stakeholders.

Production implementation

Once your project gets stakeholder commitment to move to the next phase, you can use the following iterative model to get to production. The overall lifecycle of the HDInsight project will be as shown in the following figure:



The following is a short description on what is required for each phase:

- **Executive & Stakeholder Buy-in:** Ensure you have buy-in from executives and have all requirements captured and documented
- **Discovery & Analysis:** Evaluate current data source, integrations, and pain points
- **Design:** Design the target architecture with HDInsight
- **Implementation:** This is the build or the development phase
- **User Acceptance:** Demonstrate the solution to the end users and stakeholders
- **Production Operations:** Document and have an operations plan
- **Feedback, New Requirements:** Gather new enhancement requests or new system onboarding requests

Reference sites and blogs

The following are some helpful references for HDInsight:

- <http://hadoop.apache.org/>
- <http://www.microsoft.com/BigData>
- <http://hortonworks.com/download/>
- <http://www.windowsazure.com/en-us/manage/services/hdinsight/get-started-hdinsight/>
- <http://microsoft.com/BI>
- <http://blogs.msdn.com/b/cindygross/>
- http://sqlblog.com/blogs/lara_rubbelke
- <http://blogs.msdn.com/b/carlnol/>
- <http://dennyglee.com/>
- <http://www.hdinsightblog.com/>
- <http://anindita9.wordpress.com/>
- <http://blogs.msdn.com/b/avkashchauhan/>

Summary

Hadoop Elephant has grown in size since 2007; it is now mature and ready for enterprise adoption, and HDInsight makes adoption, management, and integration easier.

There is a lot going on in this space that will further ease the adoption of Hadoop; notably Hadoop 2.0 with data federation, MapReduce NextGen YARN, HDInsight general availability, and SQL Server 2014 CTP1.

For a successful implementation, start with a good plan, build a modular architecture, and iterate through a set of requirements to show value and progress. Good luck!!!

Index

A

ad hoc queries

- about 91
- hive fort 90
- Hive query execution 91
- reference tables, creating 90

Amazon Elastic MapReduce

- URL 10

analytics functions

- in HiveQL 92

Apache Hadoop, concepts

- about 6
- cluster layout 8
- core components 7
- distributions 10, 11
- eco system 9
- HDInsight distribution, differentiators 11

Apache Hive wiki site

- URL 92

architectural considerations

- about 95
- extensible and modular 95, 96
- integration, strategy 96
- metadata-driven solution 96
- security 96

Azure

- data, shipping to 61

Azure HDInsight Service

- considerations 38
- for Windows 37

Azure management portal

- URL 47

Azure storage

- integration 45

Azure Storage Explorer

- URL 58

Azure Storage Vault (ASV)

- Azure Storage Explorer 58
- storage access keys, getting 57
- storage tools 58
- used, for loading data 57

B

Big Data

- use cases 5, 6

blogs

- references 98

business intelligence tools 92

C

Cloudera

- URL 10

Cloud Storage Studio

- URL 58

CloudXplorer

- URL 58

cluster

- deleting 46
- HDFS, accessing 41
- job results, viewing 43
- monitoring 44
- provisioning 38-40
- removing 46
- restoring 47
- sample MapReduce job, deploying 42, 43
- sample MapReduce job, executing 42, 43
- status 50
- verifying 41

cluster layout, Hadoop

- DataNode 8
- JobTracker 8
- NameNode 8
- Secondary NameNode 8
- TaskTracker 8

code, MapReduce

- compiling 71
- packaging 71

core components, Hadoop

- about 7
- Hadoop Distributed File System (HDFS) 7
- Hadoop MapReduce 7

D

data

- importing, to Excel 86-89
- loading, Azure Storage Vault (ASV) used 57
- loading, Hadoop commands used 55
- loading, interactive JavaScript used 60
- loading, Sqoop used 62
- shipping, to Azure 62

Datameer

- URL 93

DataNode 8

distribution, HDInsight

- Active Directory, integration with 11
- Analytics, Excel used 11
- differentiators 11
- Enterprise-ready Hadoop 11
- JavaScript console 11
- .NET and JavaScript, integration with 11
- RDBMS, connectors to 11
- Scale, cloud offering used 11

distributions, Hadoop

- about 10
- Amazon Elastic MapReduce, URL 10
- Cloudera, URL 10
- EMC PivitolHD, URL 11
- Hortonworks HDP, URL 11
- Microsoft HDInsight, URL 11

driver code, MapReduce 70

E

eco system, Hadoop

- about 9
- data access 9
- data, processing 10
- data stores 10
- management and integration software 10

EMC PivitolHD

- URL 11

Enterprise Data Warehouse (EDW) 62

Enterprise-ready Hadoop 11

Excel

- about 83
- data, importing 86-89
- Hive ODBC Data Source, creating 84, 85
- Hive ODBC driver, installing 84

F

files

- uploading, to blob storage 59

Flume 10

H

Hadoop

- and HDInsight, relationship 23, 24
- architectural considerations 95
- integration, strategy 96
- metadata-driven solution 96
- security 96

Hadoop client

- connecting to 55

Hadoop commands

- files, getting on local storage 56
- Hadoop client, connecting to 55
- HDFS, uploading to 57
- used, for loading data 55

Hadoop Distributed File System (HDFS) 7

Hadoop MapReduce 7

Hadoop MapReduce status 50

Hadoop NameNode status 50

Hadoop project

- data analyzing, JavaScript used 20
- data analyzing, Pig used 20
- data, collecting 12-14

- data, processing 15, 19
 - data reporting, JavaScript charts used 21
 - key phases 12
 - MapReduce, building 15, 18
 - MapReduce, executing 19
 - HBase 10**
 - Hcatalog 10**
 - HDFS**
 - about 10, 50
 - browsing 51
 - upload to 57
 - HDInsight**
 - and Hadoop, relationship 23, 24
 - end-to-end solutions 11
 - management dashboard 40, 41
 - uninstalling 34
 - HDInsight node**
 - Hive, starting 73
 - Pig, starting 78
 - HDInsight project**
 - blogs, references 98
 - lifecycle 97, 98
 - sites, references 98
 - HDInsight services**
 - managing 34
 - Hive**
 - about 9, 72
 - architecture 72
 - or Pig 78
 - relationship, with Hadoop components 72
 - starting, in HDInsight node 73
 - summary table, verifying 76
 - table, creating 74
 - table creation, summary 75
 - table, holding 75
 - Hive ODBC Data Source**
 - creating 84, 85
 - Hive ODBC driver**
 - installing 84
 - HiveQL**
 - about 72
 - analytic functions 92
 - Hortonworks Data Platform (HDP for Windows) 25**
 - Hortonworks HDP**
 - URL 11
 - Hortonworks website**
 - URL 32
- ## I
- IBM Infosphere Data Explorer**
 - URL 93
 - image source**
 - URL 6
 - interactive JavaScript**
 - used, for loading data 60
- ## J
- JavaScript**
 - for analysis 92
 - for reporting 92
 - JavaScript charts**
 - used, for reporting data 21
 - JobTracker 8**
- ## K
- key files 53**
- ## M
- Mahout 9**
 - management dashboard, HDInsight 40, 41**
 - map code, MapReduce 67**
 - MapReduce**
 - about 10, 51, 67
 - code, compiling 71
 - code, packaging 71
 - design 67
 - driver code 70
 - executing 71
 - Job History 52, 53
 - map code 67
 - reduce code 69
 - results, verifying 71
 - summary 52
 - MapR, URL 11**
 - metadata-driven solution 96**
 - Microsoft HDInsight**
 - URL 11
 - Microsoft HDInsight Dashboard 50**

Microsoft SQL Server Analysis and Reporting Services (SSAS, SSRS)

URL 92

MS Excel 201 9

multinode

common time on nodes, setting 29

firewall ports, configuring 31

network, setting up 28

planning 27, 28

preparing 27, 28

remote scripting, setting up 29, 30

multinode, installation

configuring 32

installer, running 33

software, downloading 32

validating 33

N

NameNode 8

NameNode content

backing up 53

NameNode Status browser link 50

NameNode URL 50

O

on-premise

deployment, options 24

Hortonworks Data Platform (HDP for Windows) 25

platforms, supported 25

Windows HDInsight server 24

Oozie 10

Oracle BI

URL 93

P

Personal Identifiable Information (PII) 96

Pig

about 9, 77

architecture 77

or Hive 78

starting, in HDInsight node 78

Pig Grunt script

about 78

code 79

execution 80

verification 81

Platfora

URL 93

POC 97

POSIX 96

project planning

blogs, references 98

production, implementation 97, 98

Proof of Concept (POC) plan 97

sites, references 98

Proof of Concept. *See* POC

R

reduce code, MapReduce 69

reference tables

creating 90

RITA website

URL 56, 90

S

SAP BI Platform

URL 93

SAS Enterprise Miner

URL 92

SAS Visual Analytics

URL 92

Secondary NameNode 8

share nothing architecture 7

single-node install

about 25

software, downloading 25

validating 26

wizard, running 26

sites

references 98

Sqoop

about 10

benefits 62

used, for loading data 62

using, to import SQL to Hadoop 63

using, ways 62

storage access keys 57

storage account

registering 59

T

Tableau software

URL 93

table, Hive

creating 74, 75

loading 75

summary, verifying 76

TaskTracker 8, 51

transformation

file, organizing 66

objective 66

scenario 65

W

Windows Azure Explorer

URL 58

Windows HDInsight server 24

Z

Zookeeper 10



Thank you for buying
HDInsight Essentials

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

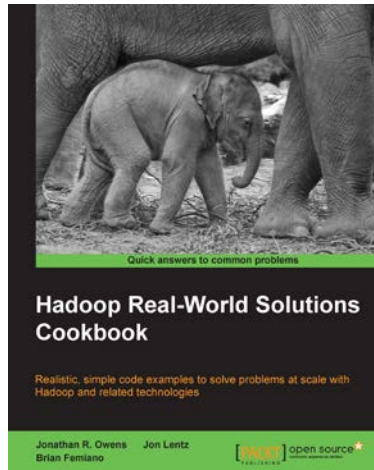
Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



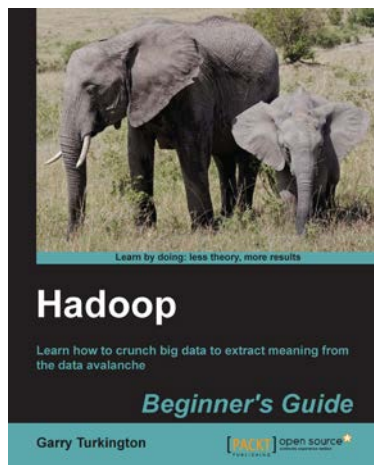
Hadoop Real-World Solutions Cookbook

ISBN: 978-1-84951-912-0

Paperback: 316 pages

Realistic, simple code examples to solve problems at scale with Hadoop and related technologies

1. Solutions to common problems when working in the Hadoop environment
2. Recipes for (un)loading data, analytics, and troubleshooting
3. In depth code examples demonstrating various analytic models, analytic solutions, and common best practices



Hadoop Beginner's Guide

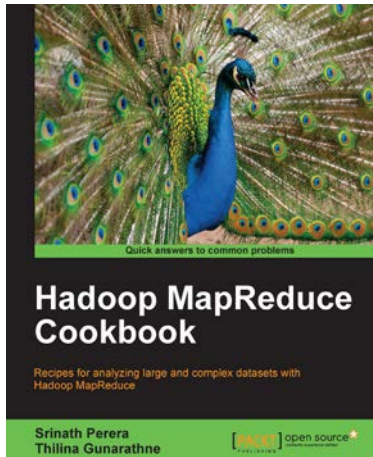
ISBN: 978-1-84951-730-0

Paperback: 398 pages

Learn how to crunch big data to extract meaning from the data avalanche

1. Learn tools and techniques that let you approach big data with relish and not fear
2. Shows how to build a complete infrastructure to handle your needs as your data grows
3. Hands-on examples in each chapter give the big picture while also giving direct experience

Please check www.PacktPub.com for information on our titles



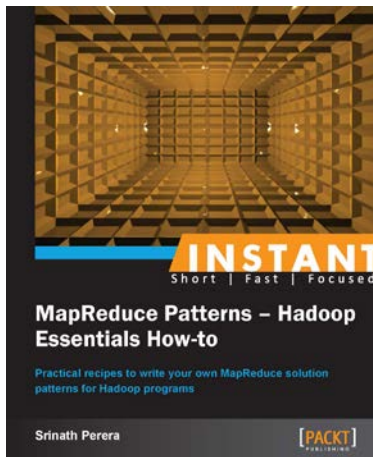
Hadoop MapReduce Cookbook

ISBN: 978-1-84951-728-7

Paperback: 300 pages

Recipes for analyzing large and complex datasets with Hadoop MapReduce

1. Learn to process large and complex data sets, starting simply, then diving in deep
2. Solve complex big data problems such as classifications, finding relationships, online marketing and recommendations
3. More than 50 Hadoop MapReduce recipes, presented in a simple and straightforward manner, with step-by-step instructions and real world examples



Instant MapReduce Patterns – Hadoop Essentials How-to

ISBN: 978-1-78216-770-9

Paperback: 60 pages

Practical recipes to write your own MapReduce solution patterns for Hadoop programs

1. Learn something new in an Instant! A short, fast, focused guide delivering immediate results.
2. Learn how to install, configure, and run Hadoop jobs
3. Seven recipes, each describing a particular style of the MapReduce program to give you a good understanding of how to program with MapReduce

Please check www.PacktPub.com for information on our titles