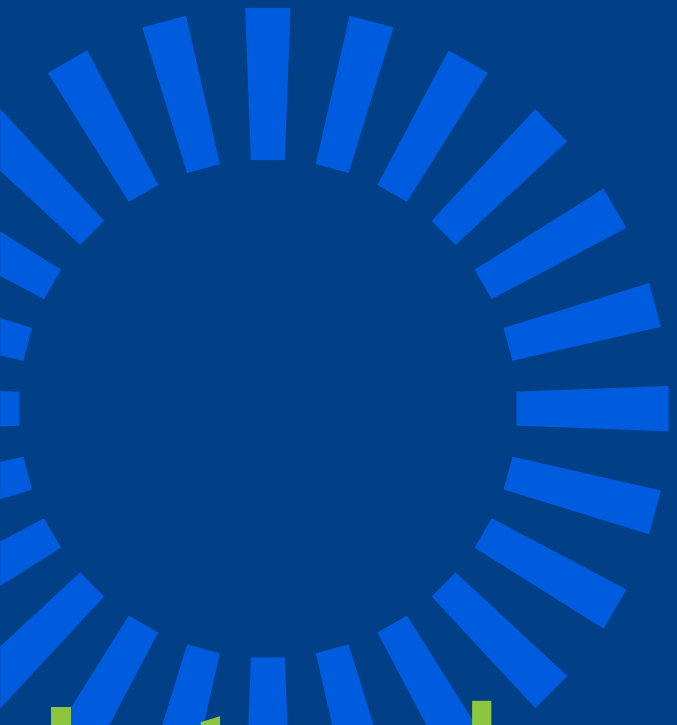


Microsoft

Introducing Microsoft Azure HDInsight

Technical Overview

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

Visit us today at

microsoftpressstore.com

- **Hundreds of titles available** – Books, eBooks, and online resources from industry experts
- **Free U.S. shipping**
- **eBooks in multiple formats** – Read on your computer, tablet, mobile device, or e-reader
- **Print & eBook Best Value Packs**
- **eBook Deal of the Week** – Save up to 60% on featured titles
- **Newsletter and special offers** – Be the first to hear about new releases, specials, and more
- **Register your book** – Get additional benefits

Microsoft

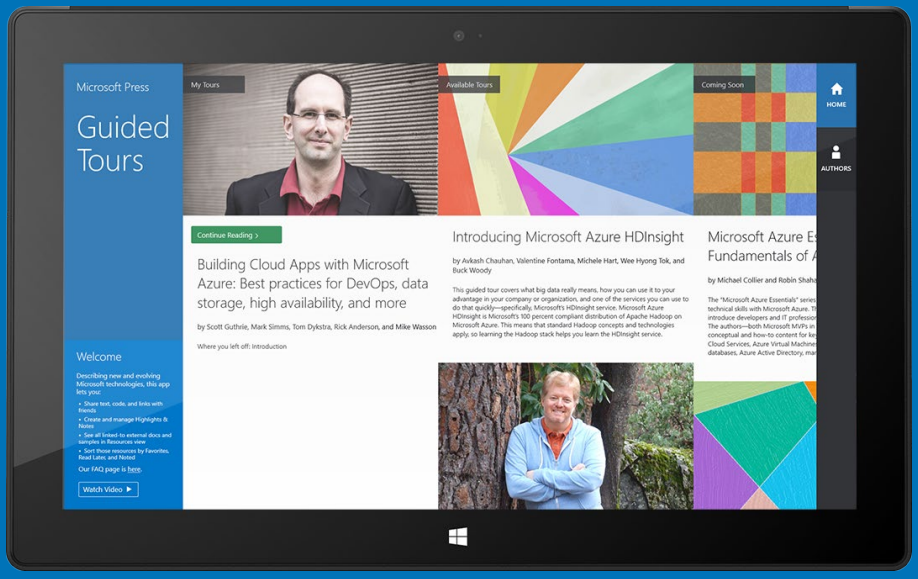
Get the latest news from Microsoft Press sent to your inbox.

- New and upcoming books
- Special offers
- Free eBooks
- How-to articles

Sign up today at MicrosoftPressStore.com/Newsletters

Microsoft

Wait, there's more...



Find more great content and resources in the **Microsoft Press Guided Tours** app.

The Microsoft Press Guided Tours app provides insightful tours by Microsoft Press authors of new and evolving Microsoft technologies.

- Share text, code, illustrations, videos, and links with peers and friends
- Create and manage highlights and notes
- View resources and download code samples
- Tag resources as favorites or to read later
- Watch explanatory videos
- Copy complete code listings and scripts

Download from Windows Store

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2014 Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

ISBN: 978-0-7356-8551-2

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft and the trademarks listed at *http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx* are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

This book expresses the authors' views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers, or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Acquisitions, Developmental, and Project Editor: Devon Musgrave
Editorial Production: Flyingspress and Rob Nance
Copyeditor: John Pierce
Cover: Twist Creative • Seattle

Table of Contents

Foreword

One could certainly deliberate about the premise that big data is a limitless source of innovation. For me, the emergence of big data in the last couple of years has changed data management, data processing, and analytics more than at any time in the past 20 years. Whether data will be the new oil of the economy and provide as significant a life-transforming innovation for dealing with data and change as the horse, train, automobile, or plane were to conquering the challenge of distance is yet to be seen. Big data offers ideas, tools, and engineering practices to deal with the challenge of growing data volume, data variety, and data velocity and the acceleration of change. While change is a constant, the use of big data and cloud technology to transform businesses and potentially unite customers and partners could be the source of a competitive advantage that sustains organizations into the future.

The cloud and big data, and in particular Hadoop, have redefined common on-premises data management practices. While the cloud has improved broad access to storage, data processing, and query processing at big data scale and complexity, Hadoop has provided environments for exploration and discovery not found in traditional business intelligence (BI) and data warehousing. The way that an individual, a team, or an organization does analytics has been impacted forever. Since change starts at the level of the individual, this book is written to educate and inspire the aspiring data scientist, data miner, data analyst, programmer, data management professional, or IT pro. HDInsight on Azure improves your access to Hadoop and lowers the friction to getting started with learning and using big data technology, as well as to scaling to the challenges of modern information production. If you are managing your career to be more future-proof, definitely learn HDInsight (Hadoop), Python, R, and tools such as Power Query and Microsoft Power BI to build your data wrangling, data munging, data integration, and data preparation skills.

Along with terms such as *data wrangling, data munging,* and *data science,* the big data movement has introduced new architecture patterns, such as data lake, data refinery, and data exploration. The Hadoop data lake could be defined as a massive, persistent, easily accessible data repository built on (relatively) inexpensive computer hardware for storing big data. The Hadoop data refinery pattern is similar but is more of a transient Hadoop cluster that utilizes constant cloud storage but elastic compute (turned on and off and scaled as needed) and often refines data that lands in another OLTP or analytics system such as a data warehouse, a data mart, or an in-memory analytics database. Data exploration is a sandbox pattern with which end users can work with developers (or use their own

development skills) to discover data in the Hadoop cluster before it is moved into more formal repositories such as data warehouses or data marts. The data exploration sandbox is more likely to be used for advance analysis—for example, data mining or machine learning—which a persistent data lake can also enable, while the data refinery is mainly used to preprocess data that lands in a traditional data warehouse or data mart.

Whether you plan to be a soloist or part of an ensemble cast, this book and its authors (Avkash, Buck, Michele Val, and Wee-Hyong) should help you get started on your big data journey. So flip the page and let the expedition begin.

Darwin Schweitzer
Aspiring data scientist and lifelong learner

Introduction

Microsoft Azure HDInsight is Microsoft's 100 percent compliant distribution of Apache Hadoop on Microsoft Azure. This means that standard Hadoop concepts and technologies apply, so learning the Hadoop stack helps you learn the HDInsight service. At the time of this writing, HDInsight (version 3.0) uses Hadoop version 2.2 and Hortonworks Data Platform 2.0.

In *Introducing Microsoft Azure HDInsight,* we cover what big data really means, how you can use it to your advantage in your company or organization, and one of the services you can use to do that quickly—specifically, Microsoft's HDInsight service. We start with an overview of big data and Hadoop, but we don't emphasize only concepts in this book—we want you to jump in and get your hands dirty working with HDInsight in a practical way. To help you learn and even implement HDInsight right away, we focus on a specific use case that applies to almost any organization and demonstrate a process that you can follow along with.

We also help you learn more. In the last chapter, we look ahead at the future of HDInsight and give you recommendations for self-learning so that you can dive deeper into important concepts and round out your education on working with big data.

Who should read this book

This book is intended to help database and business intelligence (BI) professionals, programmers, Hadoop administrators, researchers, technical architects, operations engineers, data analysts, and data scientists understand the core concepts of HDInsight and related technologies. It is especially useful for those looking to deploy their first data cluster and run MapReduce jobs to discover insights and for those trying to figure out how HDInsight fits into their technology infrastructure.

Assumptions

Many readers will have no prior experience with HDInsight, but even some familiarity with earlier versions of HDInsight and/or with Apache Hadoop and the MapReduce framework will provide a solid base for using this book. *Introducing Microsoft Azure HDInsight* assumes you have experience with web technology, programming on Windows machines, and basic

data analysis principles and practices and an understanding of Microsoft Azure cloud technology.

Who should not read this book

Not every book is aimed at every possible audience. This book is not intended for data mining engineers.

Organization of this book

This book consists of one conceptual chapter and four hands-on chapters. Chapter 1, "Big data, quick overview," introduces the topic of big data, with definitions of terms and descriptions of tools and technologies. Chapter 2, "Getting started with HDInsight," takes you through the steps to deploy a cluster and shows you how to use the HDInsight Emulator. After your cluster is deployed, it's time for Chapter 3, "Programming HDInsight." Chapter 3 continues where Chapter 2 left off, showing you how to run MapReduce jobs and turn your data into insights. Chapter 4, "Working with HDInsight data," teaches you how to work more effectively with your data with the help of Apache Hive, Apache Pig, Excel and Power BI, and Sqoop. Finally, Chapter 5, "What next?," covers practical topics such as integrating HDInsight into the rest of your stack and the different options for Hadoop deployment on Windows. Chapter 5 finishes up with a discussion of future plans for HDInsight and provides links to additional learning resources.

Finding your best starting point in this book

The different sections of *Introducing Microsoft Azure HDInsight* cover a wide range of topics and technologies associated with big data. Depending on your needs and your existing understanding of Hadoop and HDInsight, you may want to focus on specific areas of the book. Use the following table to determine how best to proceed through the book.

If you are	Follow these steps
New to big data or Hadoop or HDInsight	Focus on Chapter 1 before reading any of the other chapters.
Familiar with earlier releases of HDInsight	Skim Chapter 2 to see what's changed, and dive into Chapters 3–5.
Familiar with Apache Hadoop	Skim Chapter 1 for the HDInsight-specific content and dig into Chapter 2 to learn how Hadoop is implemented in Azure.
Interested in the HDInsight Emulator	Read the second half of Chapter 2.
Interested in integrating your HDInsight cluster into your organization	Read through first half of Chapter 5.

Book scenario

Swaddled in Sage Inc. (*Sage*, for short) is a global apparel company that designs, manufactures, and sells casual wear that targets male and female consumers 18 to 30 years old. The company operates approximately 1,000 retail stores in the United States, Canada, Asia, and Europe. In recent years, *Sage* started an online store to sell to consumers directly. *Sage* has also started exploring how social media can be used to expand and drive marketing campaigns for upcoming apparel.

Sage is the company's founder.

Natalie is Vice President (VP) for Technology for *Sage*. Natalie is responsible for *Sage's* overall corporate IT strategy. Natalie's team owns operating the online store and leveraging technology to optimize the company's supply chain. In recent years, Natalie's key focus is how she can use analytics to understand consumers' retail and online buying behaviors, discover mega trends in fashion social media, and use these insights to drive decision making within *Sage*.

Steve is a senior director who reports to Natalie. Steve and his team are responsible for the company-wide enterprise data warehouse project. As part of the project, Steve and his team have been investing in Microsoft business intelligence (BI) tools for extracting, transforming, and loading data into the enterprise data warehouse. In addition, Steve's team is responsible for rolling out reports using SQL Server Reporting Services and for building the OLAP cubes that are used by business analysts within the organization to interactively analyze the data by using Microsoft Excel.

In various conversations with CIOs in the fashion industry, Natalie has been hearing the term "big data" frequently. Natalie has been briefed by various technology vendors on the promise of big data and how big data analytics can help produce data-driven decision making within her organization. Natalie has been trying to figure out whether big data is market hype or technology that she can use to take analytics to the next level within *Sage*. Most importantly, Natalie wants to figure out how the various big data technologies can complement *Sage's* existing technology investments. She meets regularly with her management team, composed of Steve (data warehouse), Peter (online store), Cindy (business analyst), Kevin (supply chain), and Oliver (data science). As a group, the v-team has been learning from both technology and business perspectives about how other companies have implemented big data strategies.

To kick-start the effort on using data and analytics to enable a competitive advantage for *Sage*, Natalie created a small data science team, led by Oliver, who has a deep background in mathematics and statistics. Oliver's team is tasked with "turning the data in the organization into gold"—insights that can enable the company to stay competitive and be one step ahead.

One of the top-of-mind items for Oliver and Steve is to identify technologies that can work well with the significant Microsoft BI investments that the company has made over the years. Particularly, Oliver and Steve are interested in Microsoft big data solutions, as using those solutions would allow their teams to take advantage of familiar tools (Excel, PowerPivot, Power View, and, more recently, Power Query) for analysis. In addition, using these solutions will allow the IT teams to use their existing skills in Microsoft products (instead of having to maintain a Hadoop Linux cluster). Having attended various big data conferences (Strata, SQLPASS Business Analytics Conference), Oliver and Steve are confident that Microsoft offers a complete data platform and big data solutions that are enterprise-ready. Most importantly, they see clearly how Microsoft big data solutions can fit with their existing BI investment, including SharePoint.

Join us in this book as we take you through Natalie, Oliver, and Steve's exciting journey to get acquainted with HDInsight and use Microsoft BI tools to deliver actionable insights to their peers (Peter, Cindy, and Kevin).

Conventions and features in this book

This book presents information using conventions designed to make the information readable and easy to follow.

- Step-by-step instructions consist of a series of tasks, presented as numbered steps (1, 2, and so on) listing each action you must take to complete a task.
- Boxed elements with labels such as “Note” provide additional information.
- Text that you type (apart from code blocks) appears in bold.

System requirements

You need the following hardware and software to complete the practice exercises in this book:

- A Microsoft Azure subscription (for more information about obtaining a subscription, visit azure.microsoft.com and select **Free Trial**, **My Account**, or **Pricing**)
- A computer running Windows 8, Windows 7, Windows Server 2012, or Windows Server 2008 R2; this computer will be used to submit MapReduce jobs
- Office 2013 Professional Plus, Office 365 Pro Plus, the standalone version of Excel 2013, or Office 2010 Professional Plus
- .NET SDK
- Azure module for Windows PowerShell
- Visual Studio
- Pig, Hive, and Sqoop
- Internet connection to download software and chapter examples

Depending on your Windows configuration, you might need local Administrator rights to install or configure Visual Studio and SQL Server 2008 products.

Sample data and code samples

You'll need some sample data while you're experimenting with the HDInsight service. And if we're going to offer some sample data to you, we thought we’d make it something that is "real world," something that you'd run into every day—and while we're at it, why not pick something you can implement in production immediately?

The data set we chose to work with is web logs. Almost every organization has a web server of one type or another, and those logs get quite large quickly. They also contain a gold mine of information about who visits the site, where they go, what issues they run into (broken links or code), and how well the system is performing.

We also refer to a "sentiment" file in our code samples. This file contains a large volume of unstructured data that *Sage* collected from social media sources (such as Twitter), comments posted to their blog, and focus group feedback. For more information about sentiment files, and for a sample sentiment file that you can download, see http://hortonworks.com/hadoop-tutorial/how-to-refine-and-visualize-sentiment-data/.

Working with sample data

Because we're writing about a Microsoft product, we used logs created by the web server in Windows—Internet Information Server (IIS). IIS can use various formats, but you'll commonly see the Extended Log Format from W3C (http://www.w3.org/TR/WD-logfile.html) in use. This format is well structured, has good documentation, and is in wide use. Although we focus on this format in this book, you can extrapolate from the processes and practices we demonstrate with it for any web log format.

In Chapter 4, we provide a link to a sample web log used in that chapter's examples. If you have a web server, you can use your own data as long as you process the fields you have included. Of course, don't interfere with anything you have in production, and be sure there is no private or sensitive data in your sample set. You can also set up a test server or virtual machine (VM), install a web server, initialize the logs, and then write a script to hit the server from various locations to generate real but nonsensitive data. That's what the authors of this book did.

You can also mock up a web log using just those headers and fields. In fact, you can take the small sample (from Microsoft's documentation, available here: http://msdn.microsoft.com/en-us/library/ms525807(v=vs.90).aspx) and add in lines with the proper fields by using your favorite text editor or word processor:

```
#Software: Internet Information Services 6.0 #Version: 1.0 #Date: 2001-05-02 17:42:15
#Fields: time c-ip cs-method cs-uri-stem sc-status cs-version 17:42:15 172.16.255.255
GET
default.htm 200 HTTP/1.0
```

In general, the W3C format we used is a simple text file that has the following basic structure:

- **#Software** Name of the software that created the log file. For Windows, you'll see Internet Information Services followed by the IIS numbers.
- **#Version** The W3C version number of the log file format.
- **#Date** The date and time the log file was created. Note that these are under the control of the web server settings. They can be set to create multiple logs based on time, dates, events, or sizes. Check with your system administrator to determine how they set this value.
- **#Fields** Tells you the structure of the fields used in the log. This is also something the administrator can change. In this book, we're using the defaults from an older version of Windows Server, which include:
  - Time of entry
  - TCP/IP address of the client
  - HTML method called
  - Object called
  - Return code
  - Version of the web return call method

Using the code samples

Chapter 3 and Chapter 4 include sample Windows PowerShell scripts and C# code that you use to work with HDInsight. Download the code samples from http://aka.ms/IntroHDInsight/CompContent.

Acknowledgments

We'd like to thank the following people for their help with the book:

Avkash: I would like to dedicate this book to my loving family, friends, and coauthors, who provided immense support to complete this book.

Buck: I would like to thank my fellow authors on this work, who did an amazing amount of "heavy lifting" to bring it to pass. Special thanks to Devon Musgrave, who's patience is biblical. And, of course, to my wonderful wife, who gives me purpose in everything I do.

Michele: I would like to thank my children, Aaron, Camille, and Cassie-Cassandra, for all the games of run-the-bases, broom hockey and wii obstacle course; all the baking of cookies; and all the bedtime stories that I had to miss while working on this book.

Val: I would like to thank my wife, Veronica, and my lovely kids for supporting me through this project. It would not be possible without them, so I deeply appreciate their patience. Special thanks to my amazing coauthors—Wee-Hyong Tok, Michele Hart, Buck Woody, and Avkash Chauhan—and our editor Devon Musgrave for sharing in this labor of love.

Wee-Hyong: Dedicated to Juliet, Nathaniel, Siak-Eng, and Hwee-Tiang for their love, support, and patience.

Errata & book support

We've made every effort to ensure the accuracy of this book. If you discover an error, please submit it to us via mspinput@microsoft.com. You can also reach the Microsoft Press Book Support team for other support via the same alias. Please note that product support for Microsoft software and hardware is not offered through this address. For help with Microsoft software or hardware, go to http://support.microsoft.com.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: http://twitter.com/MicrosoftPress.

Chapter 1

Big data, quick intro

These days you hear a lot about *big data*. It's the new term du jour for some product you simply must buy—and buy *now*. So is big data really a thing? Is *big* data different from the regular-size, medium-size, or jumbo-size data that you deal with now?

No, it isn't. It's just bigger. It comes at you faster and from more locations at one time, and you're being asked to keep it longer. Big data *is* a real thing, and you're hearing so much about it because systems are now capable of collecting huge amounts of data, yet these same systems aren't always designed to handle that data well.

A quick (and not so quick) definition of terms

Socrates is credited with the statement "*The beginning of wisdom is a definition of terms*," so let's begin by defining some terms. Big data is commonly defined in terms of exploding data *volume*, increases in data *variety*, and rapidly increasing data *velocity*. To state it more succinctly, as Forrester's Brian Hopkins noted in his blog post "Big Data, Brewer, and a Couple of Webinars" (http://bit.ly/qTz69N): "*Big data: techniques and technologies that make handling data at extreme scale economical.*"

Expanding on that concept are the four Vs of extreme scale: volume, velocity, variety, and variability.

- **Volume** The data exceeds the physical limits of vertical scalability, implying a scale-out solution (vs. scaling up).
- **Velocity** The decision window is small compared with the data change rate.
- **Variety** Many different formats make integration difficult and expensive.
- **Variability** Many options or variable interpretations confound analysis.

Typically, a big data opportunity arises when a solution requires you to address more than one of the Vs. If you have only one of these parameters, you may be able to use current technology to reach your goal.

For example, if you have an extreme volume of relationally structured data, you could separate the data onto multiple relational database management system (RDBMS) servers. You could then query across all the systems at once—a process called "sharding."

If you have a velocity challenge, you could use a real-time pipeline feature such as Microsoft SQL Server StreamInsight or another complex event processing (CEP) system to process the data as it is transmitted from its origin to its destination. In fact, this solution is often the most optimal if the data needs to be acted on immediately, such as for alerting someone based on a sensor change in a machine that generates data.

A data problem involving variety can often be solved by writing custom code to parse the data at the source or destination. Similarly, issues that involve variability can often be addressed by code changes or the application of specific business rules and policies.

These and other techniques address data needs that involve only one or two of the parameters that large sets of data have. When you need to address multiple parameters—variety and volume, for instance—the challenge becomes more complicated and requires a new set of techniques and methods.

Use cases, when and why

So how does an organization know that it needs to consider an approach to big data? It isn't a matter of simply meeting one of the Vs described earlier, it's a matter of needing to deal with several of them at once. And most often, it's a matter of a missed opportunity —the organization realizes the strategic and even tactical advantages it could gain from the data it has or could collect. Let's take a look at a couple of examples of how dealing with big data made a real impact on an organization.

Way back in 2010, Kisalay Ranjan published a list of dozens of companies already working with large-scale data and described the most powerful ways they were using that data. In the years since, even more companies and organizations have started leveraging the data they collect in similar and new ways to enable insights, which is the primary goal for most data exercises.

A lot of low-hanging-fruit use cases exist for almost any organization:

- Sentiment analysis
- Website traffic patterns
- Human resources employee data

- Weather correlation effects
- Topographic analysis
- Sales or services analysis
- Equipment monitoring and data gathering

These use cases might not apply to every business, but even smaller companies can use large amounts of data to coordinate sales, hiring, and deliveries and support multiple strategic and tactical activities. We've only just begun to tap into the vast resources of data and the insights they bring.

These are just a few of the areas in which an organization might have a use case, but even when an organization does identify opportunities, its current technology might not be up to the task of processing it. So although it isn't a use case for big data, the use case for Hadoop, and by extension HDInsight, is to preprocess larger sets of data so that downstream systems can deal with them. At Microsoft we call this "making big rocks out of little rocks."

Tools and approaches—scale up and scale out

For the case of extreme scale, or data volume, an inflection point occurs where it is more efficient to solve the challenge in a distributed fashion on commodity servers rather than increase the hardware inside one system. Adding more memory, CPU, network capacity, or storage to handle more compute cycles is called *scale up*. This works well for many applications, but in most of these environments, the system shares a common bus between the subsystems it contains. At high levels of data transfer, the bus can be overwhelmed with coordinating the traffic, and the system begins to block at one or more subsystems until the bus can deal with the stack of data and instructions. It’s similar to a checkout register at a grocery store: an efficient employee can move people through a line faster but has to wait for the items to arrive, the scanning to take place, and the customer to pay. As more customers arrive, they have to wait, even with a fast checkout, and the line gets longer.

This is similar to what happens inside a RDBMS or other single-processing data system. Storage contains the data to be processed, which is transferred to memory and computed by the CPU. You can add more CPUs, more memory, and a faster bus, but at some point the data can overwhelm even the most capable system.

Another method of dealing with lots of data is to use more systems to process the data. It seems logical that if one system can become only so large, that adding more systems makes the work go faster. Using more systems in a solution is called *scale out*, and this approach is used in everything from computing to our overcrowded grocery store. In the case of the grocery store, we simply add more cashiers and the shoppers split themselves evenly (more or less) into the available lanes. Theoretically, the group of shoppers checks out and gets out of the store more quickly.

In computing it's not quite as simple. Sure, you can add more systems to the mix, but unless the software is instructed to send work to each system in an orderly way, the additional systems don't help the overall computing load. And there's another problem—the data. In a computing system, the data is most often stored in a single location, referenced by a single process. The solution is to distribute not only the processing but the data. In other words, move the processing to the data, not just the data to the processing.

In the grocery store, the "data" used to process a shopper is the prices for each object. Every shopper carries the data along with them in a grocery cart (we're talking precomputing-era shopping here). The data is carried along with each shopper, and the cashier knows how to process the data the shoppers carry—they read the labels and enter the summations on the register. At the end of the evening, a manager collects the computed results from the registers and tallies them up.

And that's exactly how most scale-out computing systems operate. Using a file system abstraction, the data is placed physically on machines that hold a computing program, and each machine works independently and in parallel with other machines. When a program completes its part of the work, the result is sent along to another program, which combines the results from all machines into the solution—just like at the grocery store. So in at least one way, not only is the big data problem not new, neither is the solution!

Hadoop

Hadoop, an open-source software framework, is one way of solving a big data problem by using a scale-out "divide and conquer" approach. The grocery store analogy works quite well here because the two problems in big data (moving the processing to the data and then combining it all again) are solved with two components that Hadoop uses: the Hadoop Distributed File System (HDFS) and MapReduce.

It seems that you can't discuss Hadoop without hearing about where it comes from, so we'll spend a moment on that before we explain these two components. After all, we can't let you finish this book without having some geek credibility on Twitter!

From the helpful article on Hadoop over at Wikipedia:

> Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project (http://en.wikipedia.org/wiki/Hadoop#History).

Hadoop is a *framework*, which means that it is composed of multiple components and is constantly evolving. The components in Hadoop can work separately, and often do. Several other projects also use the framework, but in this book we'll stick with those components available in (and to) the Microsoft Azure HDInsight service.

HDFS

The Hadoop Distributed File System (HDFS) is a Java-based layer of software that redirects calls for storage operations to one or more nodes in a network. In practice, you call for a file object by using the HDFS application programming interface (API), and the code locates the node where the data is located and returns the data to you.

That's the short version, and, of course, it gets a little more complicated from there. HDFS can replicate the data to multiple nodes, and it uses a *name node* daemon to track where the data is and how it is (or isn't) replicated. At first, this was a single point of failure, but later releases added a secondary function to ensure continuity.

So HDFS allows data to be split across multiple systems, which solves one problem in a large-scale data environment. But moving the data into various places creates another problem. How do you move the computing function to where the data is?

MapReduce

The Hadoop framework moves the computing load out to the data nodes through the use of a MapReduce paradigm. MapReduce refers to the two phases of distributed processing: a *map* phase in which the system determines where the nodes are located, moving the work to those nodes, and a *reduce* phase, where the system brings the intermediate results back together and computes them. Different engines implement these functions in different ways, but this loose definition will work for this chapter, and we'll refine it in later chapters as you implement the code in the various examples.

Hadoop uses a *JobTracker* process to locate the data and transfer the compute function and a *TaskTracker* to perform the work. All of this work is done inside a Java Virtual Machine (JVM).

You can read a great deal more about the technical details of the Apache Hadoop project here: http://hadoop.apache.org/docs/current/.

HDInsight

The HDInsight service is a type of implementation of Hadoop that runs on the Microsoft Azure platform. Working with Hortonworks, Microsoft worked to properly license and source the code and contributes back to the Hadoop source project. HDInsight is 100 percent compatible with Apache Hadoop because it builds on the Hortonworks Data Platform (HDP).

You could, of course, simply deploy virtual machines running Windows or one of several distributions of Linux on Azure and then install Hadoop on those. But the fact that Microsoft implements Hadoop as a service has several advantages:

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

As mentioned, the HDInsight service runs on Microsoft Azure, and that requires a little explaining before we proceed further.

Microsoft Azure

Microsoft Azure isn't a product, it's a *series* of products that form a complete cloud platform, as shown in Figure 1-1. At the very top of the stack in this platform are the data centers

where Azure runs. These are modern data centers, owned and operated by Microsoft using the Global Foundation Services (GFS) team that runs Microsoft properties such as Microsoft.com, Live.com, Office365.com, and others. The data centers are located around the world in three main regions: the Americas, Asia, and Europe. The GFS team is also responsible for physical and access security and for working with the operating team to ensure the overall security of Azure. Learn more about security here: http://azure.microsoft.com/en-us/support/trust-center/security/.

The many products and features within the Microsoft Azure platform work together to allow you to do three things, using various models of computing:

- **Write software** Develop software on site using .NET and open-source languages, and deploy it to run on the Azure platform at automatic scale.
- **Run software** Install software that is already written, such as SQL Server, Oracle, and SharePoint, in the Azure data centers.
- **Use software** Access services such as media processing (and, of course, Hadoop) without having to set up anything else.

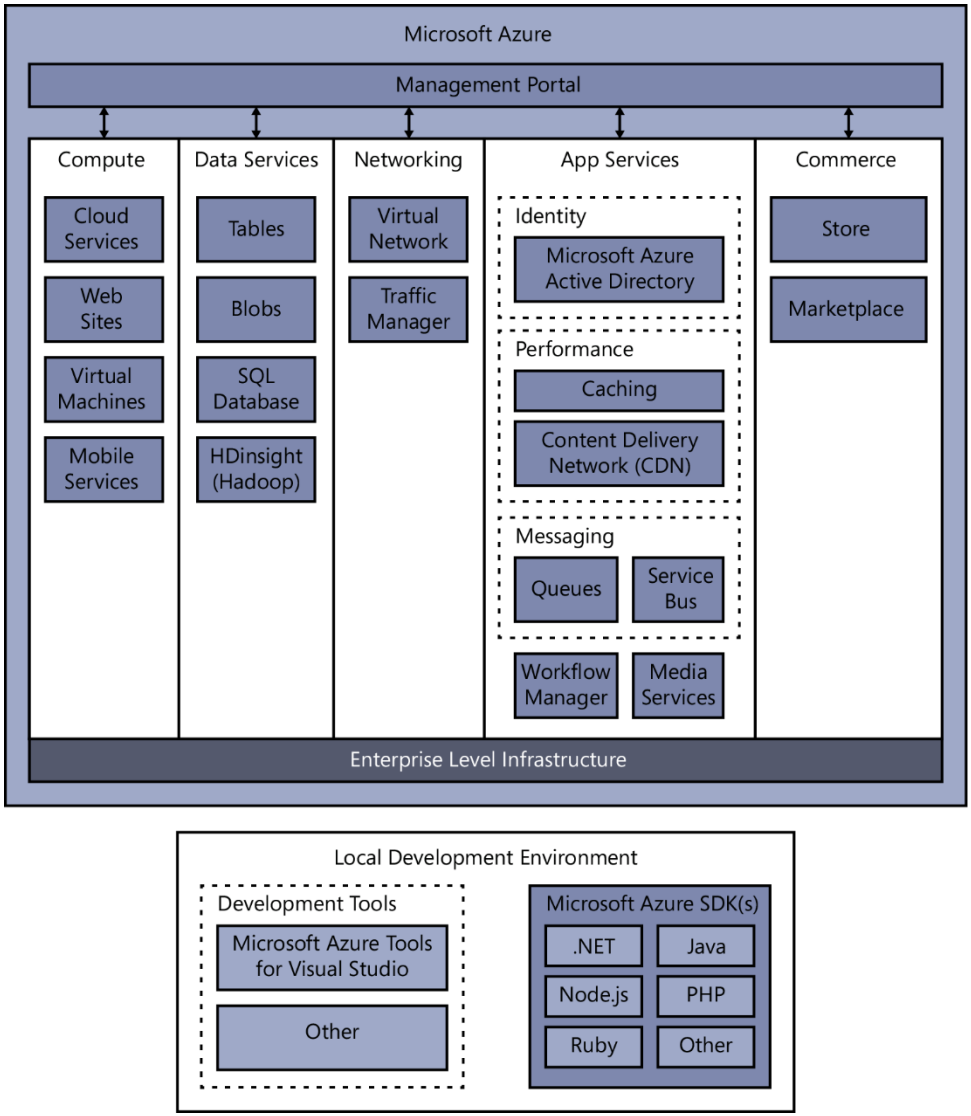


FIGURE 1-1 Overview of the Microsoft Azure platform.

Services

Microsoft Azure started within a Platform as a Service, or PaaS model. In this model of distributed computing (sometimes called the *cloud*), you write software on your local system

using a software development kit (SDK), and when you're done, you upload the software to Azure and it runs there. You can write software using any of the .NET languages or open-source languages such as JavaScript and others. The SDK runs on your local system, emulating the Azure environment for testing (only from the local system), and you have features such as caching, auto-scale-out patterns, storage (more on that in a moment), a full service bus, and much more. You're billed for the amount of services you use, by time, and for the traffic out of the data center. (Read more about billing here: http://www.windowsazure.com/en-us/pricing/details/hdinsight/.) The PaaS function of Azure allows you to write code locally and run it at small or massive scale—or even small *to* massive scale. Your responsibility is the code you run and your data; Microsoft handles the data centers, the hardware, and the operating system and patching, along with whatever automatic scaling you've requested in your code.

You can also use Azure to run software that is already written. You can deploy (from a portal, code, PowerShell, System Center, or even Visual Studio) virtual machines (VMs) running the Windows operating system and/or one of several distributions of Linux. You can run software such as Microsoft SQL Server, SharePoint, Oracle, or almost anything that will run inside a VM environment. This is often called *Infrastructure as a Service* or *IaaS*. Of course, in the IaaS model, you can also write and deploy software as in the PaaS model—the difference is the distribution of responsibility. In the IaaS function, Microsoft handles the data centers and the hardware. You're responsible for maintaining the operating system and the patching, and you have to figure out the best way to scale your application, although there is load-balancing support at the TCP/IP level. In the IaaS model, you're billed by the number of VMs, the size of each, traffic out of the machine, and the time you keep them running.

The third option you have on the Azure platform is to run software that Microsoft has already written. This is sometimes called *Software as a Service* or *SaaS*. This term is most often applied to services such as Microsoft Office 365 or Live.com. In this book we use SaaS to refer to a service that a technical person uses to further process data. It isn't something that the general public would log on to and use. The HDInsight service is an example of SaaS—it's a simple-to-deploy cluster of Hadoop instances that you can use to run computing jobs, and when your computations are complete, you can leave the cluster running. turn it off, or delete it. The cost is incurred only while your cluster is deployed. We'll explore all this more fully in a moment.

It's important to keep in mind that although all of the services that the Azure platform provides have unique capabilities and features, they all run in the same data center and can call each other seamlessly. That means you could have a PaaS application talking to a smartphone that uses storage that an internal system loads data into, process that data

using a VM in IaaS, and then process that data further in the HDInsight service and send along the results to yet another system using the service bus. You could just as easily use only the storage feature to retain backups from an onsite system for redundancy. Everything in Azure works together or as a standalone feature.

Storage

Although you can use any of the components in Azure to perform almost any computing task, the place to start for using the HDInsight service is with storage. You'll need a place to load your data and optionally a place to store the results. Let's explore the basics of Azure Storage and then cover the specifics of how HDInsight accesses it.

The HDInsight service can actually access *two* types of storage: HDFS (as in standard Hadoop) and the Azure Storage system. When you store your data using HDFS, it's contained within the nodes of the cluster and it must be called through the HDFS API. When the cluster is decommissioned, the data is lost as well. The option of using Azure Storage provides several advantages: you can load the data using standard tools, retain the data when you decommission the cluster, the cost is less, and other processes in Azure or even from other cloud providers can access the data.

Azure Storage comes in three basic types: blobs (binary storage), tables (key/value pair storage, similar to NoSQL architectures), and queues. Interestingly, a queue is a type of table storage, so there are technically *two* types of storage in Azure. All storage is kept within a container, which you create in a particular data center. Inside the container you can create a blob or a table. By default, Azure replicates all data three times within a data center for internal redundancy and can optionally replicate all copies to a separate geographical location.

Blobs are further divided into two types: *block* and *page*. In general, you won't have to care about this distinction because, by default, HDInsight makes the decision about which type to use when it creates the data. The primary difference between the two is that block blobs are optimized for working with large files over a network, and page blobs are optimized for more random reads and writes. For the HDInsight service, you store data as blobs. In Chapter 2, "Getting started with HDInsight," you'll learn more about setting up your Azure account, creating your storage account, and loading data for processing. You can create the storage ahead of time or let the HDInsight setup process do that for you. If you create the storage account first, you should always save it in the same data center where you intend to create your HDInsight cluster.

HDInsight service

The Azure HDInsight service is a fully integrated Apache Foundation Hadoop software project. This means that standard Hadoop concepts and technologies apply, so learning the Hadoop stack helps you learn the HDInsight service.

To use the HDInsight service to run a job to do some work, you create your cluster, select the size, and access the Azure blob data you've loaded to your account. You can deploy the cluster from the Microsoft Azure portal (Figure 1-2) or use PowerShell scripts to set it all up.

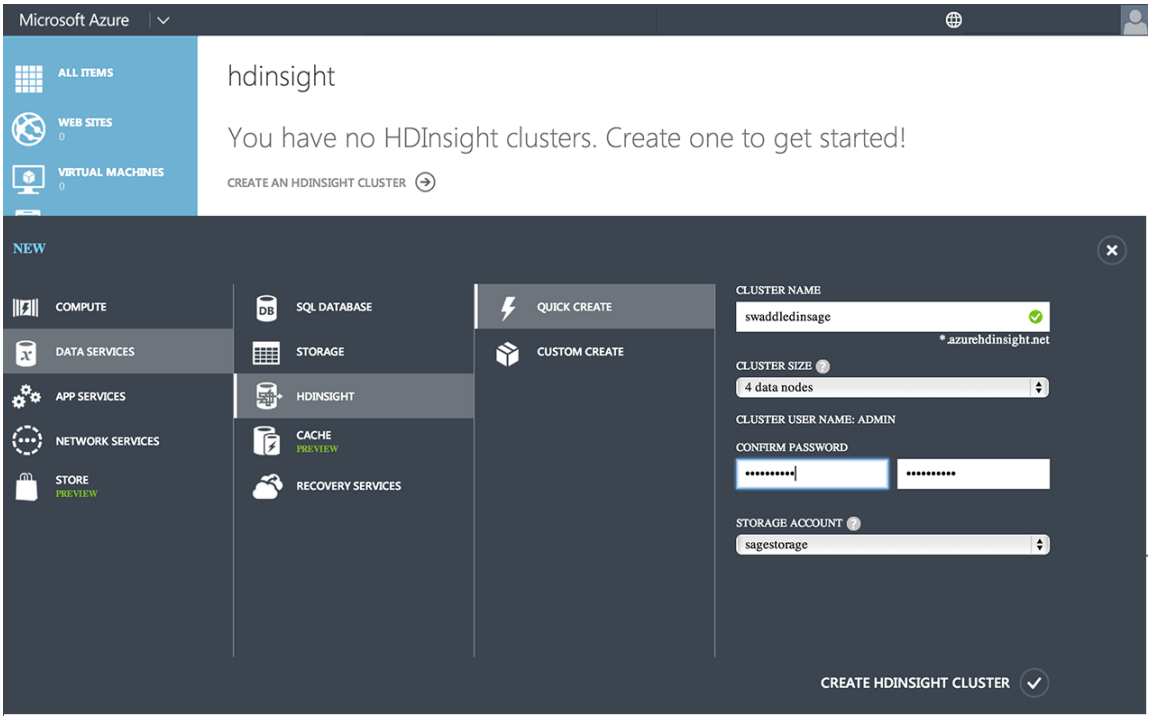


FIGURE 1-2 Use the Azure portal to create an HDInsight cluster.

After the cluster is set up, you can use the Azure portal or PowerShell to submit and control jobs that run your code (more on that in the following chapters) from your desktop. When the jobs are complete, you can remove the cluster (and stop paying for it) via PowerShell.

Note Not familiar with PowerShell? You can find a quick introduction to it here: http://technet.microsoft.com/en-us/library/bb978526.aspx.

Of course, there's a bit more to HDInsight than just deploying it and running code. The *real* work is in submitting the jobs that HDInsight can run, and to do that you can work with one or more interfaces to the system.

Interfaces

Once you've sourced your data, loaded it into Azure Storage, and deployed a cluster on the HDInsight service, you need to set up the work you want HDInsight to do. You have a few options, from running jobs directly on the cluster to using other programs that connect to the cluster to perform the calculations. In this book, our use case includes using Pig, Hive, .NET, Java, and Excel.

Pig

One program you have available is a set of software called Pig. Pig is another framework (developers love creating frameworks). It implements the Pig Latin programming language, but many times you'll see both the platform and the language simply referred to as Pig. Using the Pig Latin language, you can create MapReduce jobs more easily than by having to code the Java yourself—the language has simple statements for loading data, storing it in intermediate steps, and computing the data, and it's MapReduce aware. The official site for the Pig language is https://pig.apache.org/. You'll see an example of using Pig in Chapter 4, "Working with HDInsight data."

Hive

When you want to work with the data on your cluster in a relational-friendly format, you can use a software package called Hive. Hive allows you to create a data warehouse on top of HDFS or other file systems and uses a language called *HiveQL*, which has a lot in common with the Structured Query Language (SQL). Hive uses a metadata database, and we'll describe the process to work with it in more detail in Chapter 4**.** You can learn more about Hive at https://hive.apache.org/.

Other interfaces and tools

In addition to Pig and Hive, numerous other programs, interfaces, and tools are available for working with your HDInsight cluster. In this book, we use .NET and Java in Chapter 3, "Programming HDInsight." *Sqoop* (*not* implemented as another framework) can load data to and from a RDBMS such as SQL Server or the Azure SQL Database. Sqoop is detailed at https://sqoop.apache.org/, and we cover it later in the book as well.

Although you can look at the data using multiple products, we can't leave out one of the world's most widely distributed data-analysis tools: Excel. That's right, Microsoft has engineered the HDInsight service to accept commands from Excel and return data so that you and your users can visualize, augment, combine, and calculate the results with other data in a familiar environment. We'll cover that process in more depth in Chapter 4.

HDInsight Emulator

But you don't have to deploy your jobs or even stand up a cluster on Azure to develop and test your code. Microsoft has an HDInsight Emulator that runs locally on your development system. Using this emulator, you're able to simulate the HDInsight environment in Azure locally and test and run your jobs before you deploy them. Of course, there are some differences and considerations for working in this environment, which we cover in more depth in Chapter 2.

Summary

When you add up all of the ways you can load data, process it with HDInsight interfaces, and then visualize it with any number of tools (from Microsoft and other vendors as well), you end up with a complete stack of technologies for working with big data—from its source all the way to viewing the data in a spreadsheet—without having to leave the Microsoft ecosystem. Of course, you're not locked in, either. You can use lots of technologies to work with the data at any stage of the ingress, processing, and viewing stages.

Now let's take that sample data set referenced in the intro and get started!

Chapter 2

Getting started with HDInsight

In this chapter, we explain how to get started, from setting up your Microsoft Azure account through loading some data into your account and setting up your first cluster. We also show you how to deploy and manage an HDInsight cluster, even using a free or low-cost account for testing and learning, and cover the on-premises testing emulator.

HDInsight as cloud service

Microsoft Azure is a cloud service provided by Microsoft, and one of the services offered is HDInsight, an Apache Hadoop-based distribution running in the cloud on Azure. Another service offered by Azure is blob storage. Azure Storage functions as the default file system and stores data in the cloud and not on-premises or in nodes. This way, when you are done running an HDInsight job, the cluster can be decommissioned (to save you money) while your data remains intact.

The combination of Azure Storage and HDInsight provides an ultimate framework for running MapReduce jobs. MapReduce is the Hadoop data-processing framework used for parallel processing of data using multiple nodes.

Creating an HDInsight cluster is quick and easy: log in to Azure, select the number of nodes, name the cluster, and set permissions. The cluster is available on demand, and once a job is completed, the cluster can be deleted but the data remains in Azure Storage. Having the data securely stored in the cloud before, after, and during processing gives HDInsight an edge compared with other types of Hadoop deployments. Storing the data this way is particularly useful in cases where the Hadoop cluster does not need to stay up for long periods of time. It is worth noting that some other usage patterns, such as the data exploration pattern (also known as the data lake pattern), require the Hadoop cluster and the data to be persisted at all times. In these usage patterns, users analyze the data directly on Hadoop, and for these cases, other Hadoop solutions, such as the Microsoft Analytics Platform System or Hortonworks Data Platform for Windows, are more suitable.

Microsoft Azure subscription

Before *Swaddled in Sage* can begin using HDInsight, the company needs an Azure subscription. A subscription has many moving parts, and http://azure.microsoft.com has interactive pricing pages, a pricing calculator, and plenty of documentation to help *Swaddled in Sage* make the right selections for its usage requirements and budget.

Once the company has an Azure subscription, it can choose an Azure service and keep it running for as long as it likes. The company will be charged based on its usage and type of subscription. You can find more about billing at http://www.windowsazure.com/en-us/pricing/details/hdinsight/.

Swaddled in Sage has years' worth of sales, inventory, and customer data in a data warehouse. The company is interested in seeing just how easy it really is to provision a cluster that includes the HDInsight service and the Storage service. Oliver, a member of the team exploring HDInsight, decides to sign up for the free trial and use Microsoft's sample data to run a quick job in the cloud, save the results to Azure Storage, and then decommission the cluster. This is the model that Oliver is interested in: uploading *Swaddled in Sage* data to Microsoft Azure cloud storage—where it will remain safe even when he decommissions the cluster— minimizing hardware expense, and paying for a cluster only when he needs it.

Open the Azure Management Portal

The Azure Management Portal is a great place for Oliver to start. From the portal, he can create, configure, and manage his trial cluster and storage account, as well as other Microsoft Azure services and applications. To open the portal, he starts at azure.microsoft.com, clicks **Portal**, and logs in.

The left pane of the portal displays the list of services available, including HDInsight and Storage. Clicking a service reveals additional information about that service, including actions you can take and the service's status.

Because Oliver hasn't yet enabled HDInsight for this Azure account, when he clicks **HDInsight**, the message "You have no HDInsight clusters" is displayed and a link is provided to create an HDInsight cluster (Figure 2-1). He'll get to that in a bit. First, he wants to create a storage account.

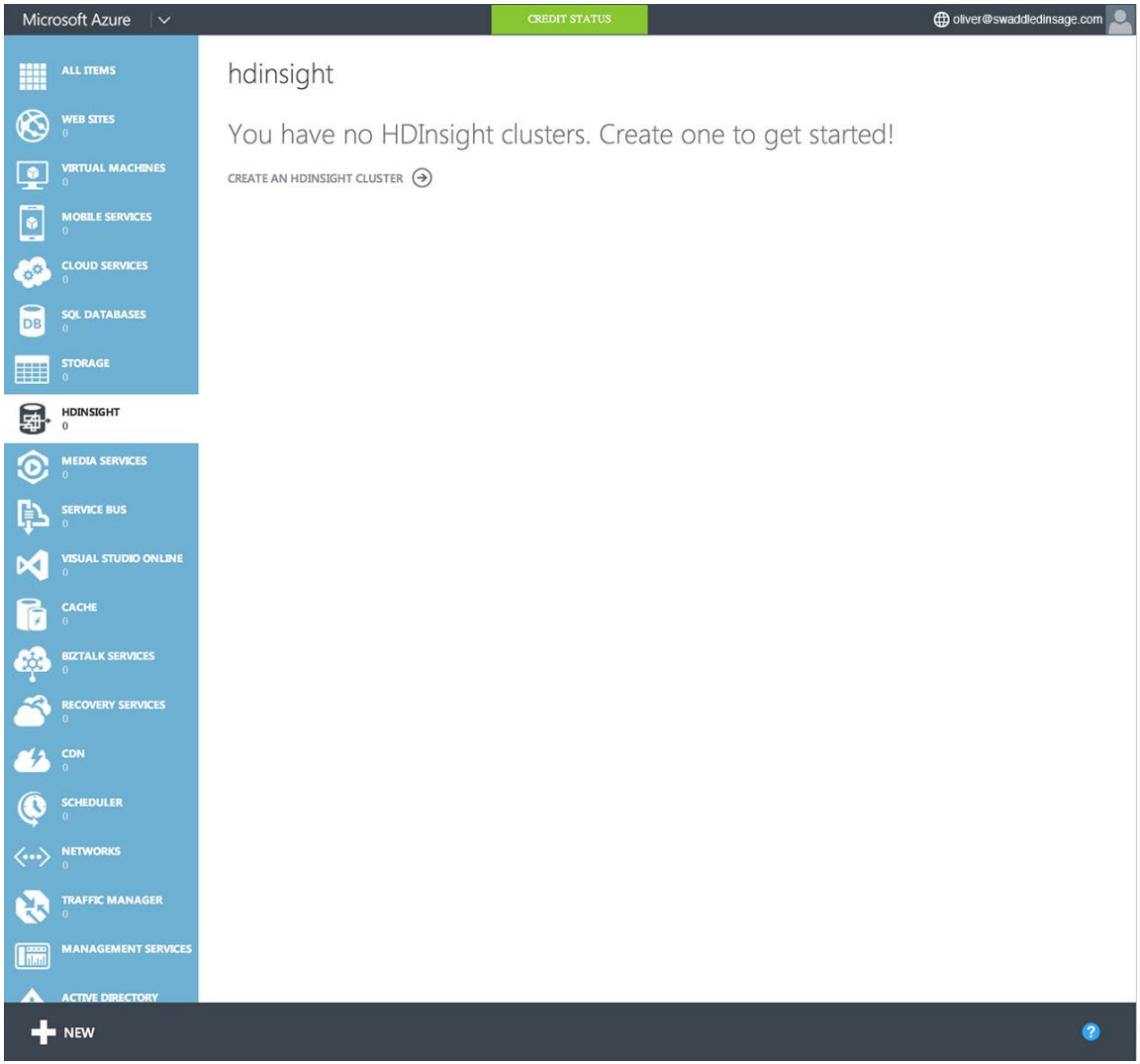


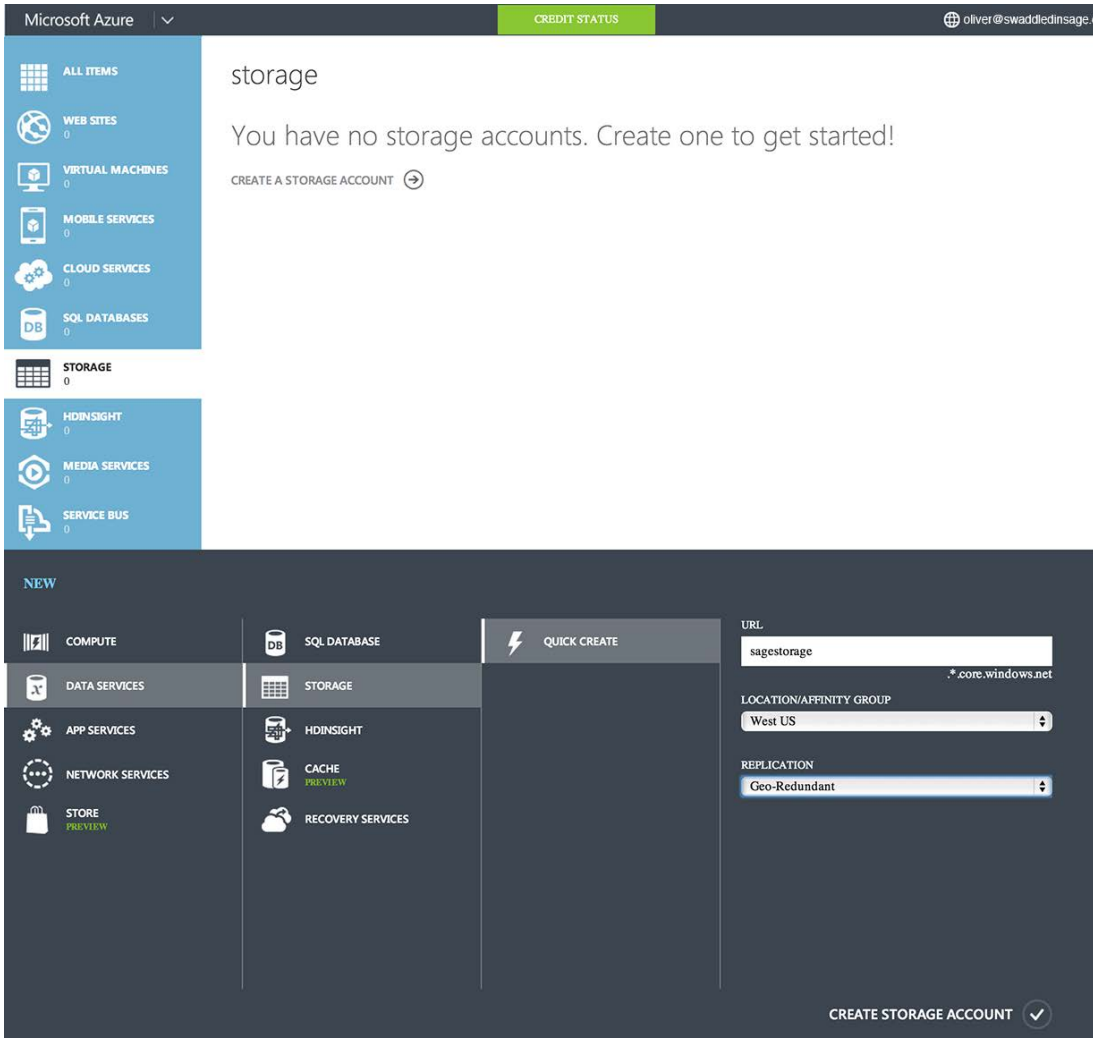
FIGURE 2-1 HDInsight in the Azure Management Portal.

Add storage to your Azure subscription

It's best for Oliver to set up a storage account before provisioning a cluster because when he sets up his cluster, he needs to tell HDInsight where the data is stored.

To create the storage account, he follows these steps:

1. In the left pane of the portal, select **Storage**.
2. At the bottom of the screen, click **New**. In the panel that opens, Data Services, Storage is already selected, as shown here.



3. Select **Quick Create**.
4. Oliver names his storage account **sagestorage.**

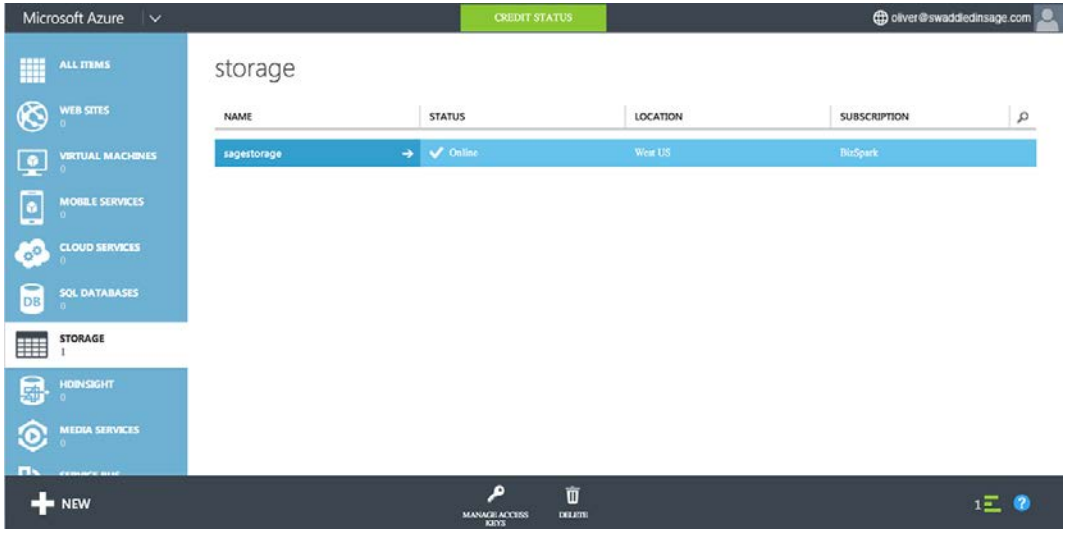
Note The URL field is deceiving; you don't actually have to enter an entire URL, just a name for your storage.

5. The closest data center is **West US**. To reduce network latency and for the best performance, Oliver stores the data as close to his location as possible.

   By selecting West US, Oliver is placing his storage account in the West US data center. But data centers can be huge. To ensure that his storage account and cluster are physically located close together in that data center, and not two city blocks away, he can create and select an affinity group. If he had previously created an affinity group, he could select that here.

6. When he uploads his own data, Oliver might want to select **Geo-Redundant** to create a backup of his data in another region. In the event of a regional disaster that takes down an entire data center, the *Swaddled in Sage* data will be available from another region.

7. Oliver clicks **Create Storage Account**, and it takes just a few minutes to set up. When the status changes to Online, as shown here, the storage account is ready for use.

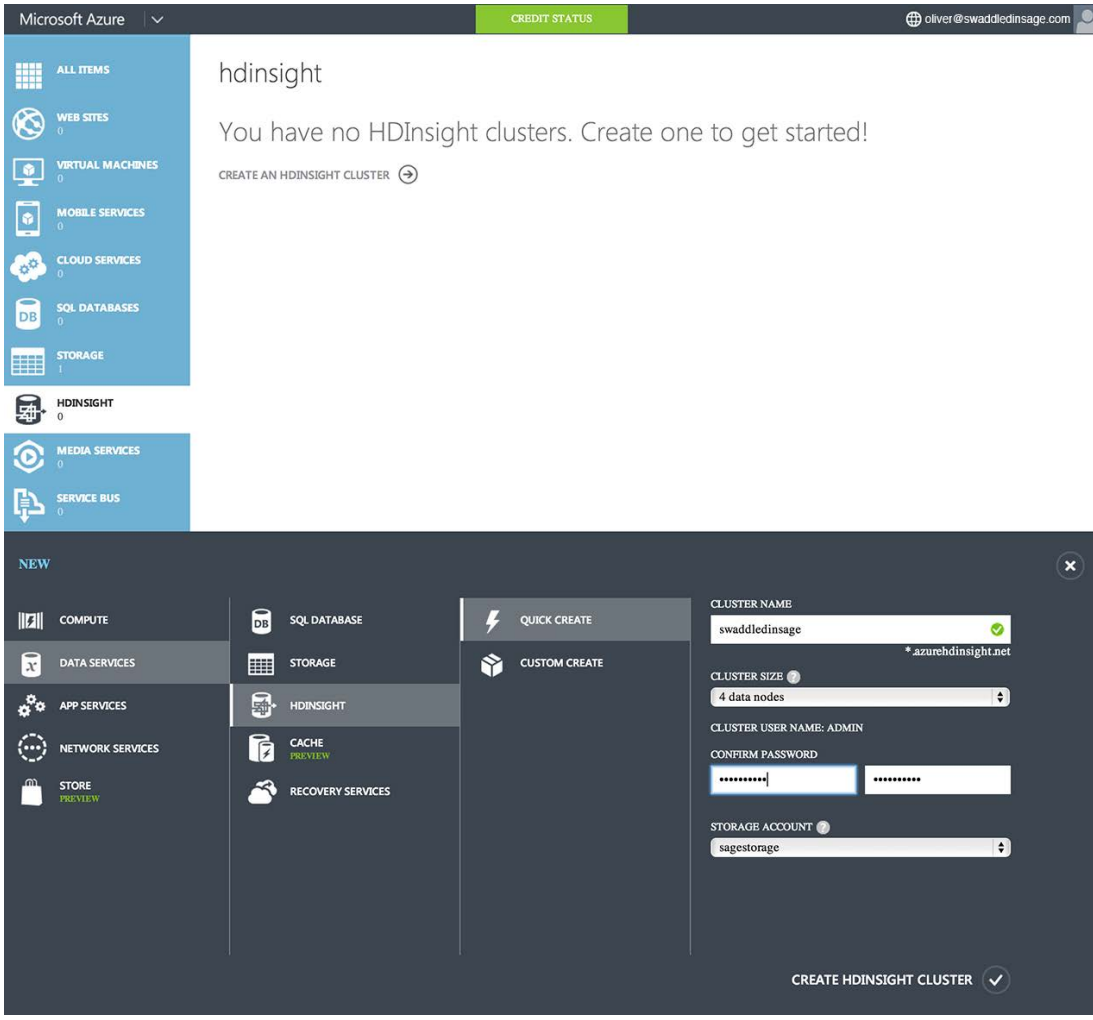


Create an HDInsight cluster

Now that Oliver has his storage account ready, it's time to create a test cluster. The process is similar to creating the storage account. Oliver starts in the portal, selects **HDInsight**, and enters the details.

1. In the left pane of the Azure Management Portal, highlight **HDInsight**.

   At the bottom of the screen, click **New**. In the panel that opens, Data Services, HDInsight is already selected.



2. Select **Quick Create**.

3. The name of the cluster becomes part of the URL for that cluster. For example, Oliver names his cluster **swaddledinsage**, so the URL for the cluster becomes swaddledinsage.azurehdinsight.net.

4. At the current time, HDInsight provides six cluster sizes: 1 node, 2 nodes, 4 nodes, 8 nodes, 16 nodes, and 32 nodes. Oliver selects 4 nodes.

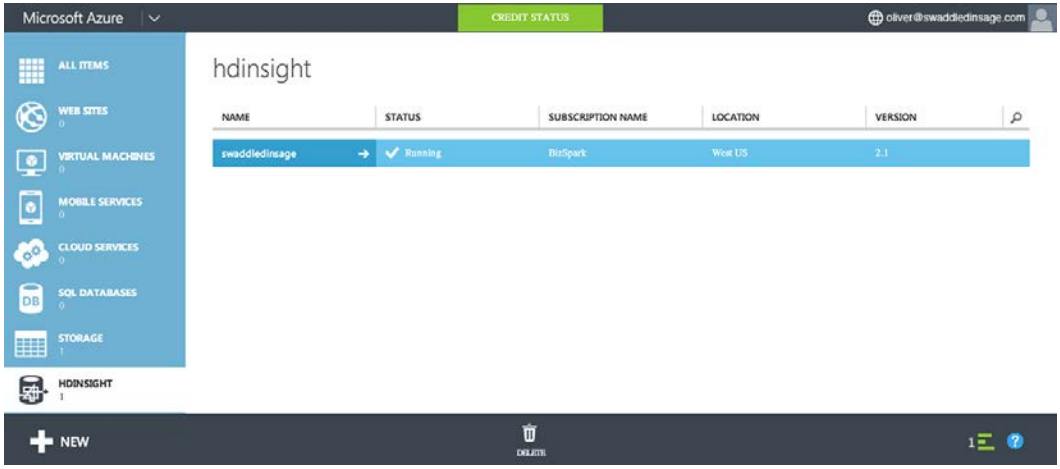
5. The next step is to create a password for accessing the cluster. This password applies

only to this cluster and must be used in conjunction with the cluster user name to access the cluster from the portal or from an outside application. Since Oliver is using Quick Create, the user name for accessing his cluster is **Admin**. He can disable or modify these access credentials later if needed. If Oliver really wants to use a different user name, he would be better off creating the cluster with the Custom Create option instead of Quick Create.

> **Note** The password must be at least 10 characters and contain at least one uppercase letter, one lowercase letter, one number, and one special character.

6. Oliver selects the sagestorage storage account he created in the earlier procedure. Since his storage is located in West US, when he selects it here, Azure creates his cluster in the same data center. HDInsight recommends that both the storage account and the cluster be located in the same data center. As mentioned earlier, the reasons for this are performance and cost. Oliver will get better performance and no data-transfer charges by co-locating his HDInsight cluster with the data.

7. When Oliver clicks **Create HDInsight Cluster**, the Microsoft Azure backend service provisions the HDInsight cluster. The amount of time required to create the cluster depends on the size and location. For Oliver, this step takes approximately 10 minutes. Oliver will know his cluster is ready when the status changes to Running. The following screenshot shows the swaddledinasage cluster online and located in West US (as is the associated sagestorage storage account shown earlier).



Manage a cluster from the Azure Management Portal

The portal is a great jumping-off point for managing and monitoring your cluster. It has three major areas: Dashboard, Monitor, and Configuration. Let's take a look at these areas one at a time.

The cluster dashboard

Selecting the name of the cluster opens its dashboard. The dashboard displays detailed information about the selected cluster, including current usage, the status of active mapper and reducer jobs, remote access, and more, as shown in Figure 2-2.

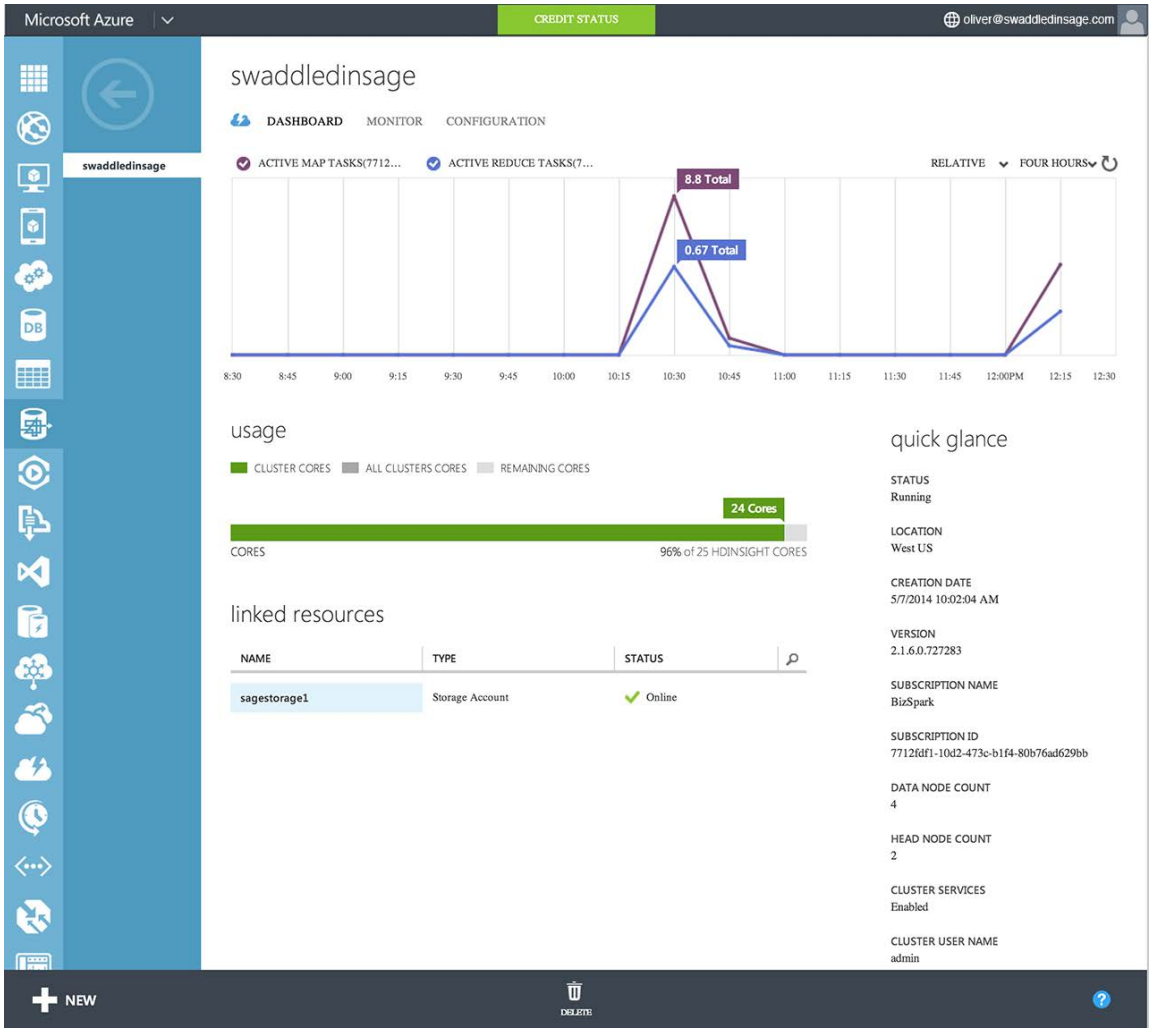


FIGURE 2-2 Cluster Dashboard view.

The number of active map and reduce tasks is visible at the top. Oliver can change the timeframe of the graph from 1 hour to 4 hours' worth of activity. Clicking the check marks next to Active Map and Active Reduce toggles the display of tasks by type.

The Usage section shows usage by cluster cores, all cluster cores, and remaining cores for our account. For Oliver's four-node cluster, 24 cores are used by default. The head node uses 8, and each worker node uses 4.

The **Linked Resources** area lists storage accounts linked to our cluster. By clicking the storage account name, Oliver can drill down into his storage containers and then to the individual blobs and files in that container.

The **Quick Glance** area displays summary information about the cluster, including its location, creation date, and number of nodes. Most of the summary information is self-explanatory, except maybe cluster services. Figure 2-2 shows that cluster services is enabled. We'll describe how to manage cluster services in the Configuration area of the portal later in this chapter.

Monitor a cluster

The Monitor area shows the currently active map and reduce tasks (these also appear in the Dashboard view) along with the shortest to longest running mapper and reducer tasks. Depending on the number of active and completed jobs, the list may have multiple pages. Since the swaddledinsage cluster is new, we ran a couple of sample jobs in the cluster to make it active. The Monitor area for the cluster is shown in Figure 2-3.

FIGURE 2-3 Cluster Monitor view.

Configure a cluster

Let's check in on Oliver and his swaddledinsage cluster. There are several useful settings that Oliver can tune from the Configuration screen. For example, this screen is where he can

enable and disable access to his cluster (Figure 2-4). If he decides to keep Hadoop Services on, he can interact with the cluster by using tools such as Windows PowerShell, the HDInsight .NET SDK, Ambari, and Oozie. And if he enables Remote Desktop, he can connect to his HDInsight name node by using a supported RDP application.

> **Note** Earlier in this chapter we mentioned the c**luster services** item in the Quick Glance section of the dashboard. The status shown for cluster services corresponds to cluster connectivity—if remote connectivity is enabled in the Configuration screen, cluster services is displayed as Enabled.

While Hadoop Services is enabled by default, Remote Desktop (RDP) access is disabled by default, and Oliver definitely wants to turn this on. The link to enable RDP access is easy to miss—it's labeled **Enable Remote** and is at the bottom of the window.

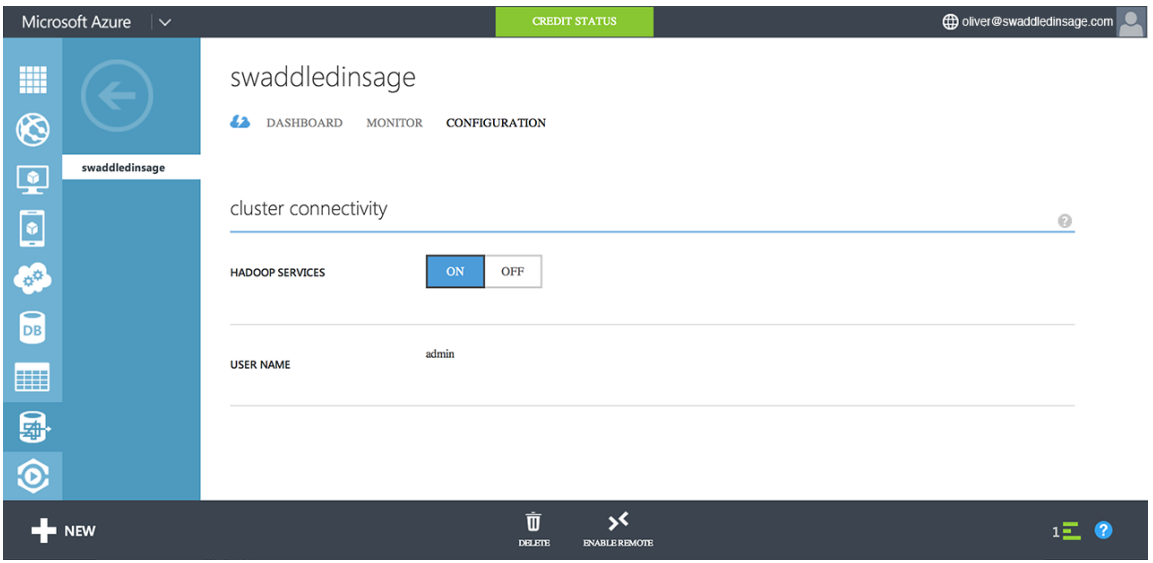


FIGURE 2-4 Cluster Configuration view.

If you make a change to cluster connectivity, remember to click **Save** (see Figure 2-5) at the bottom of the screen.

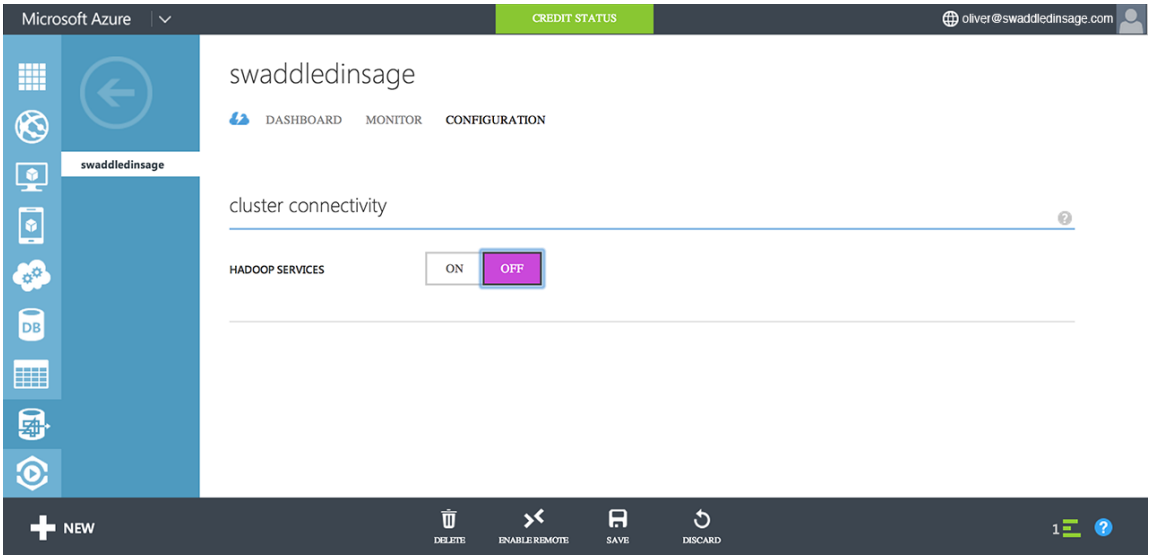
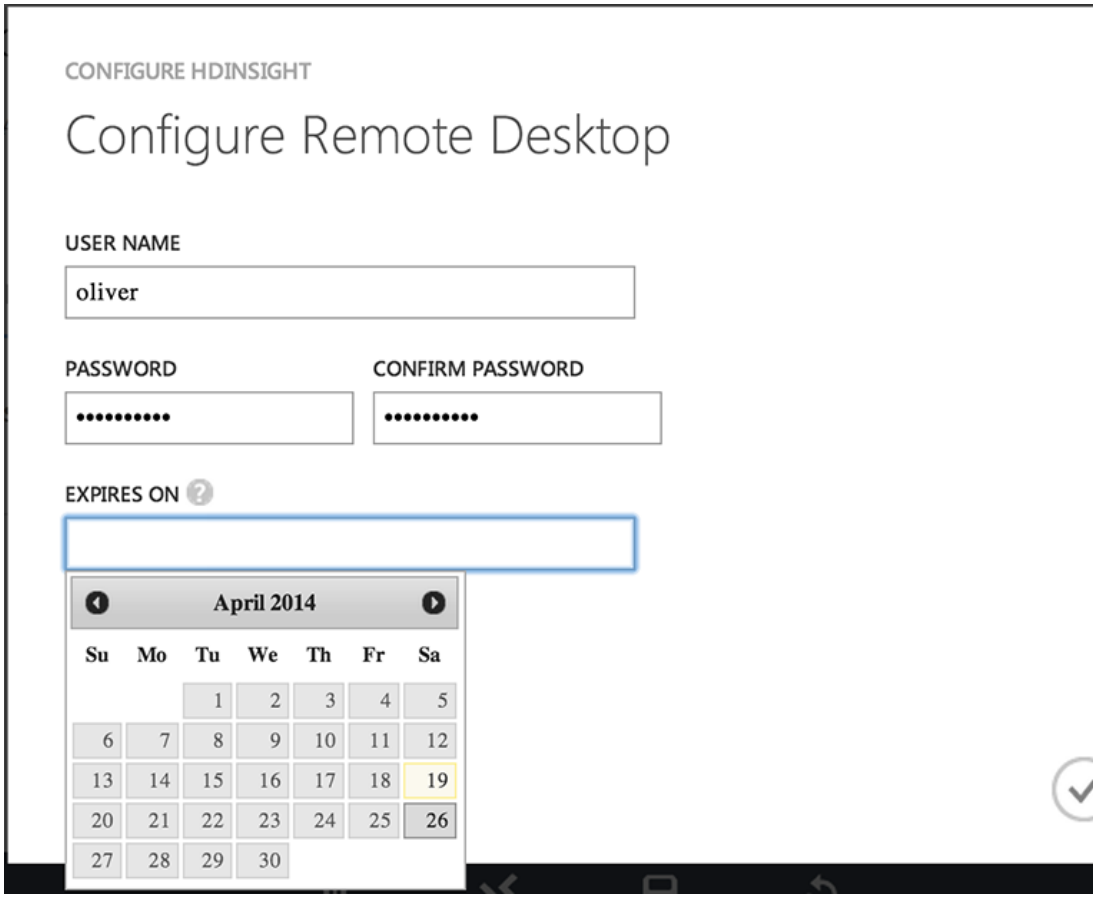


FIGURE 2-5 Save changes to cluster connectivity.

1. To enable remote access, Oliver first clicks **Enable Remote**.

2. Next, he provides a proper user name and password, as shown here. Remote desktop connection can be enabled for a maximum of seven days. Since Oliver is using the trial version to evaluate HDInsight, he limits remote desktop to one day.



When a configuration is selected and then confirmed in Azure, the bottom of the Azure Management page shows activity (Figure 2-6). Expand the bottom notification dialog box by clicking the activity icon in the bottom-right corner. The notification dialog box provides information about what actions have just completed and what other actions are in the pipeline to be completed by Microsoft Azure for the currently selected Azure service.

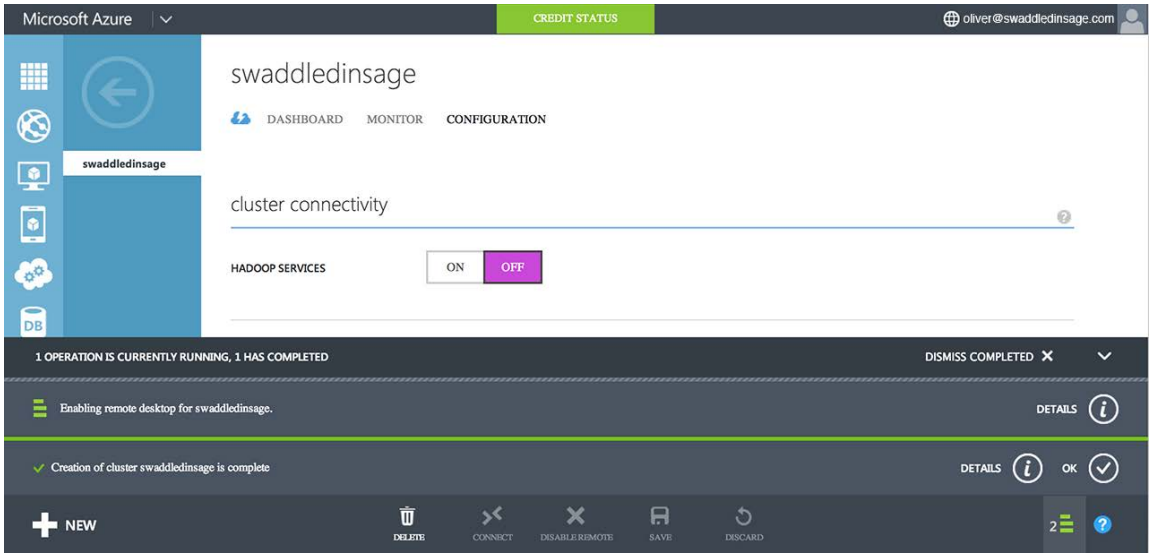


FIGURE 2-6 Cluster connectivity status changes are saved.

Accessing the HDInsight name node using Remote Desktop

In the previous section, Oliver created an HDInsight cluster in Microsoft Azure by using the Azure Management Portal. Now he'll use Remote Desktop to connect to and explore that cluster.

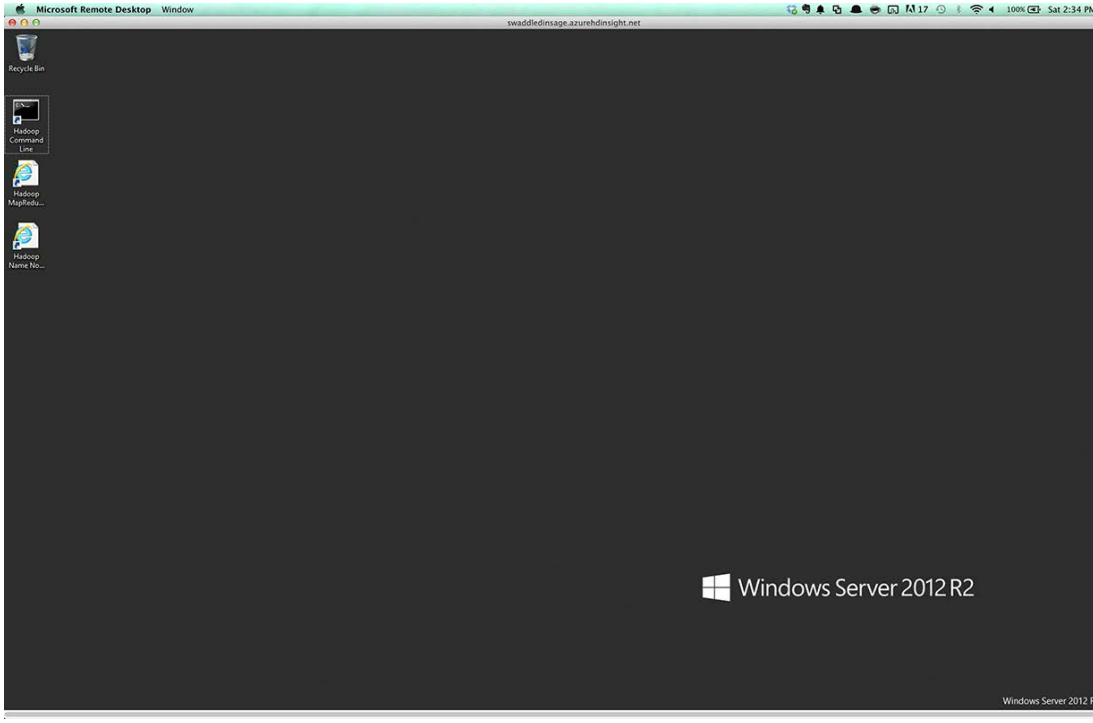
To do this, he needs to download his cluster-specific RDP file from the management portal. He'll use this file with Microsoft's Remote Desktop application or any other application that supports RDP. This file contains all the information he needs to connect to the HDInsight name node.

1. Click **Connect** at the bottom of the portal.

   Note The Connect button is active only if you've enabled remote access. For instructions, see the previous section of this chapter.

2. When prompted, Oliver opens the file to display three new shortcuts to his cluster on his desktop, shown in the following screenshot. The shortcuts are:

- Hadoop Command Line
- Hadoop MapReduce Status
- Hadoop Name Node Status



Hadoop name node status

The Hadoop data storage layer is the Hadoop Distributed File System (HDFS), and HDFS can be accessed from the management portal, the Hadoop command line, or from the Hadoop web service at port 50070.

To launch the web service, Oliver uses the Hadoop Name Node Status shortcut on his desktop. From the web interface, he can monitor the health of his cluster, view basic information about the name node and cluster, and access the file system and logs. Figure 2-7 shows that he has a four-node cluster with a capacity of 3.87 TB and about 99 percent free HDFS space.

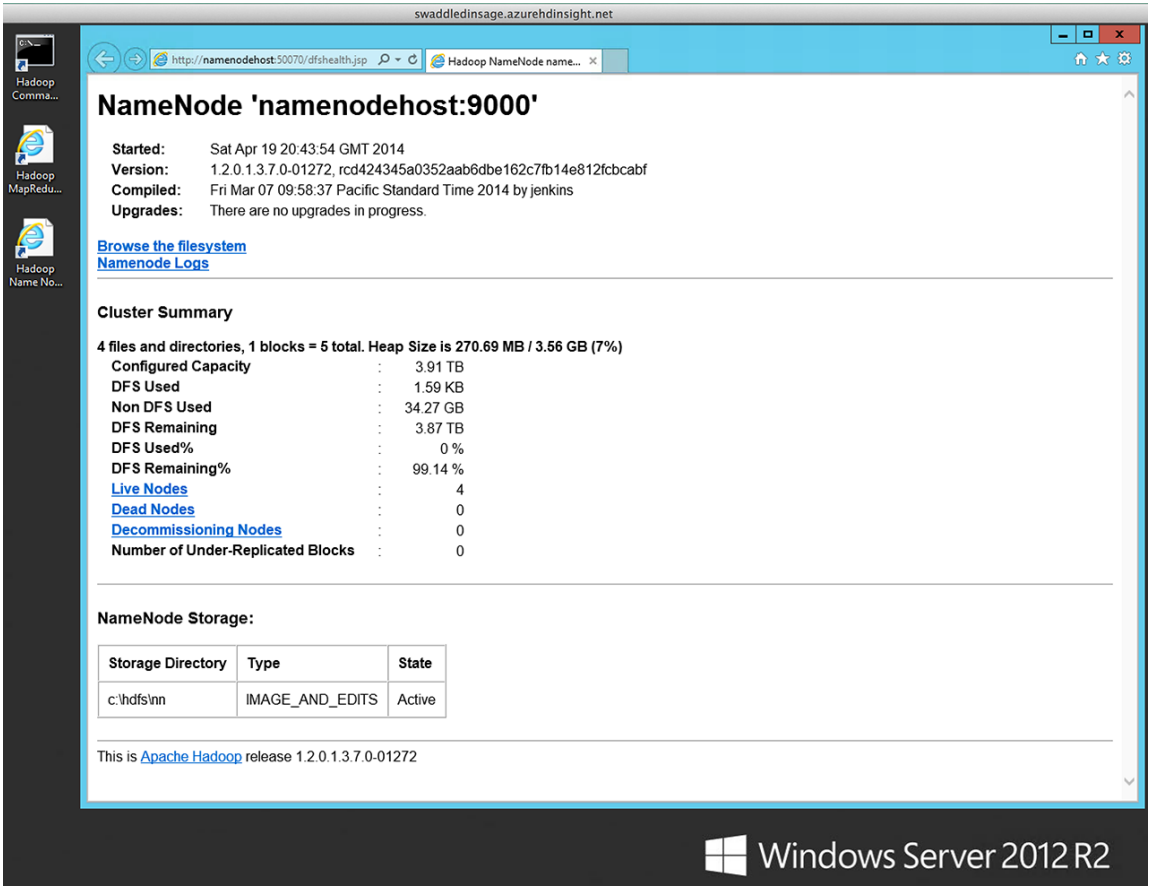
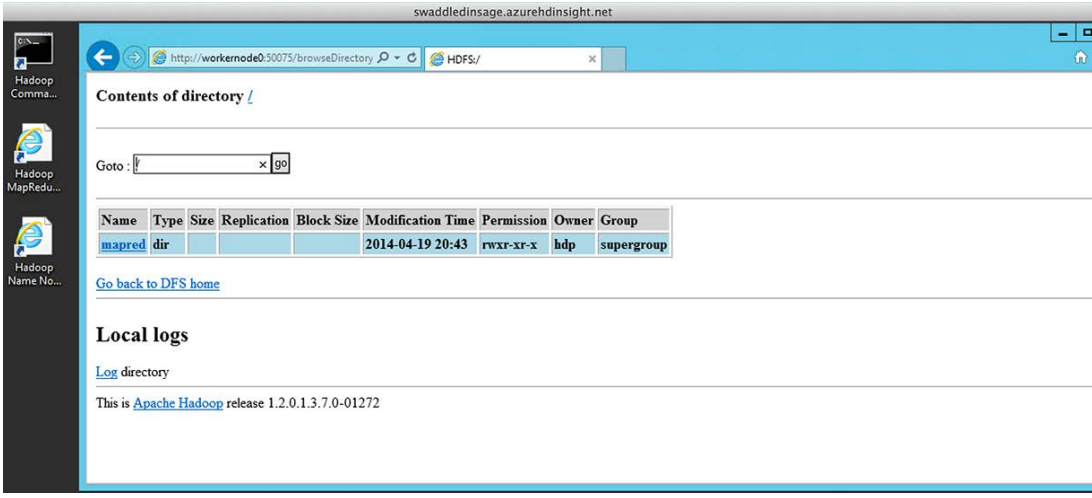


FIGURE 2-7 Hadoop name node status web service.

To access just the name node logs, Oliver can click **Namenode Logs**, but what he really wants to see are the contents of the HDFS file system, so he selects **Browse The Filesystem** and sees the information shown here.



From here, Oliver can drill down to find information about each file and folder, including permissions, owners, and group details. For the files on HDFS, he can also look up information such as total size, total number of blocks, and replication factors.

To learn more about his four nodes, Oliver selects **Live Nodes** (see Figure 2-7). As shown in Figure 2-8, each data node is currently active and has 1 TB of space configured. Each node has between five and seven blocks occupied.

There's plenty more Oliver can do from this screen, and many of those tasks are covered in later chapters.

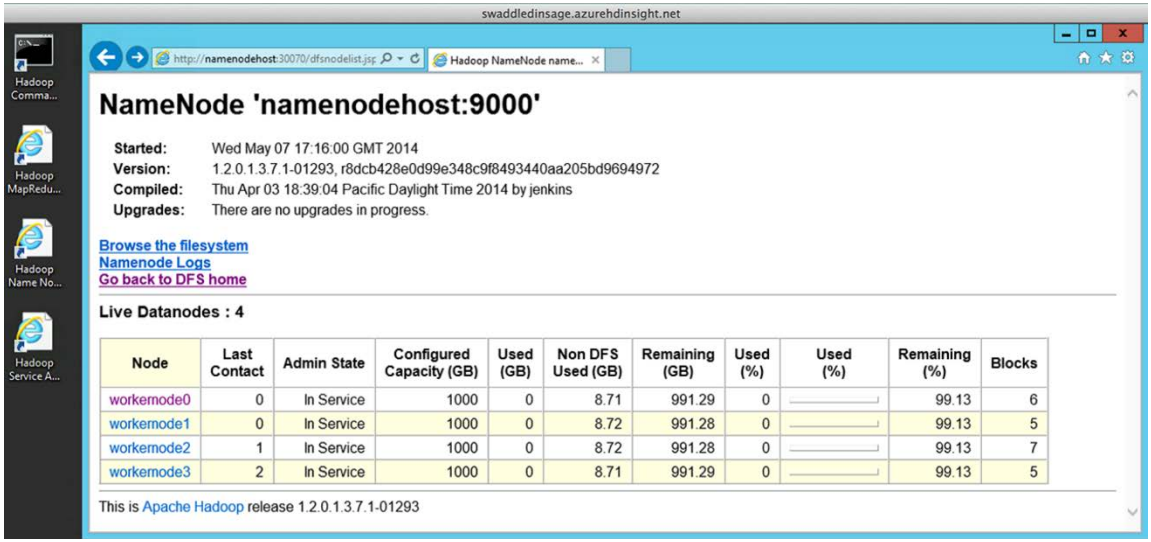


FIGURE 2-8 Live nodes.

Hadoop MapReduce status

The Hadoop data processing layer has two components, JobTracker and TaskTracker. JobTracker usually runs on the same server as the name node. However, if needed, it can also run on a different machine. The role of JobTracker is to accept the jobs submitted to an HDInsight cluster from any source, break the job into tasks, and assign those tasks to the data nodes in the cluster. TaskTracker, which runs on each data node, sends status messages to JobTracker as it works on each job. To see the status of any job running on an HDInsight cluster, use the Hadoop web server on port 50030, following the URL template *http://your_hadoop_server:50030/*, or the Hadoop MapReduce Status shortcut.

Oliver selects the Hadoop MapReduce Status shortcut, which launches a web browser and opens the Hadoop web server on port 30030 to display JobTracker details as shown in Figure 2-9. (Note that the default port for the JobTracker is 50030.)

> **Important** If you're reading this book in order, you know that the swaddledinsage cluster is new and that no scheduled jobs have been run on the cluster yet. The following screenshots show lots of activity for some sample jobs we ran. For example, Figure 2-9 shows currently running map and reduce tasks.

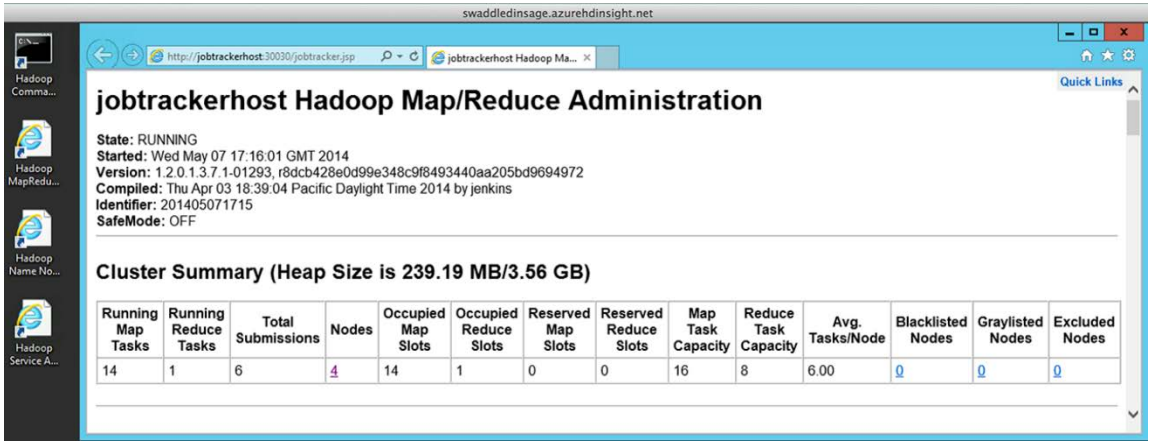


FIGURE 2-9 Hadoop MapReduce status.

The MapReduce status webpage displays information about running jobs, previously executed jobs, retired jobs, and scheduling, as shown in Figure 2-10.

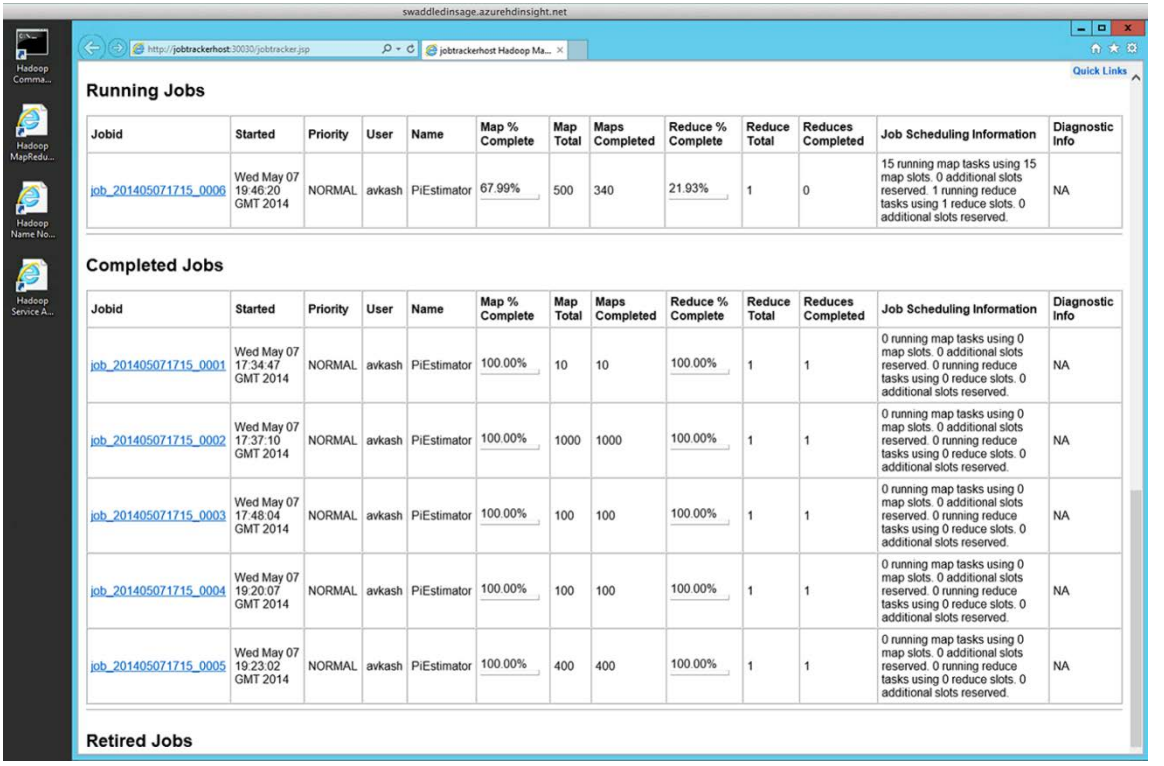


FIGURE 2-10 A list of MapReduce jobs actively running and completed.

This page has many active links, and clicking them displays more specific status information. Oliver is interested in seeing the status and performance of the actively running job, so he selects job_201405071715_0006 from the job list table and gets a detailed summary about this job as shown in Figure 2-11.

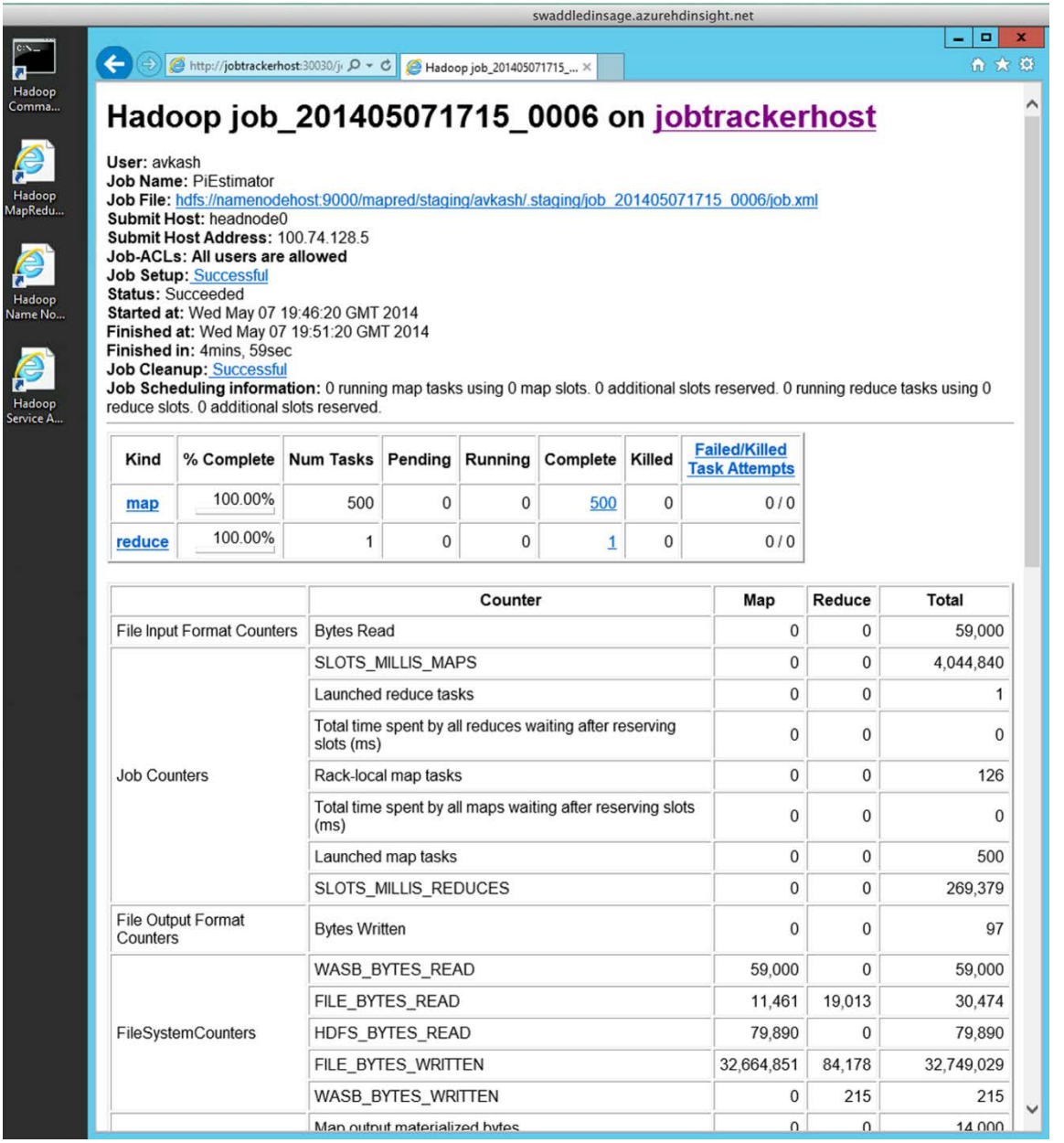


FIGURE 2-11 Detailed summary about a specific job, with job ID job_201405071715_0006.

Some of the information Oliver sees includes the following:

- Job IDs and other job-specific details for active and completed jobs
- The status of current and previously run jobs (%Complete, Running, Failed, and so on)
- Information about total mappers and reducers for the job
- Mapper and reducer counters and statistics such as disk and memory usage

To learn more about active map tasks, Oliver clicks the **Map** link and sees a list of active map tasks, as shown in Figure 2-12.

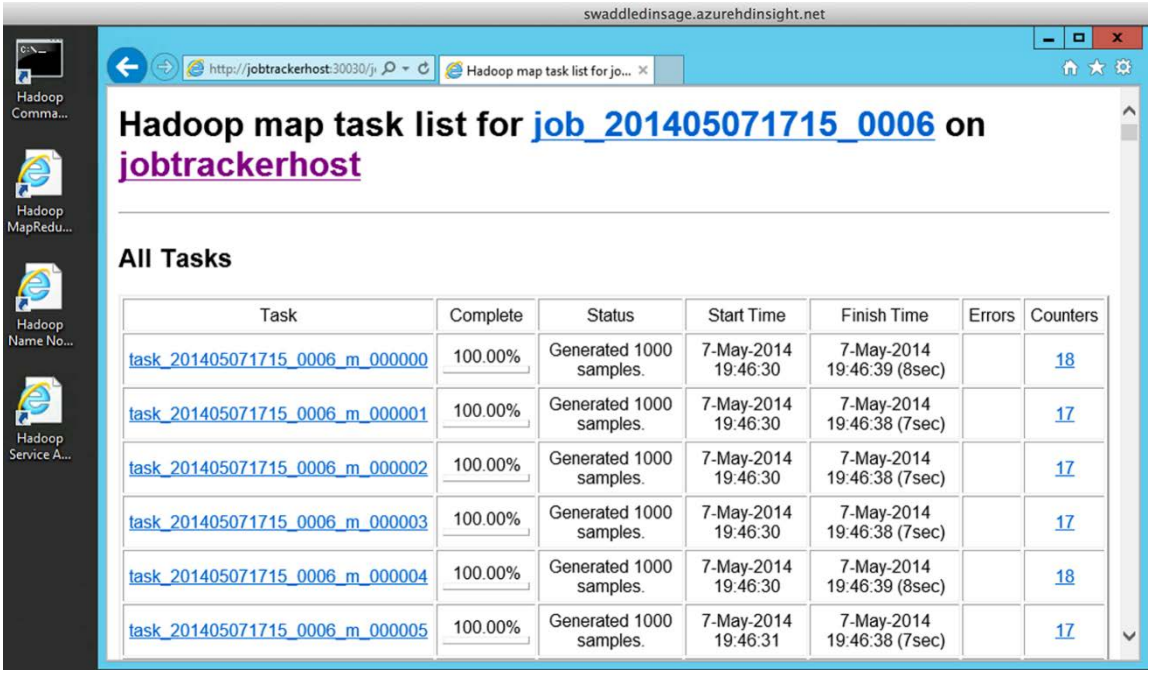


FIGURE 2-12 A list of map tasks for MapReduce job job_201405071715_0006.

To learn more about active reduce tasks, Oliver clicks the **Reduce** link (see Figure 2-11) and sees a list of active reduce tasks (Figure 2-13). Only one reduce task is active and is 100 percent complete with no errors.

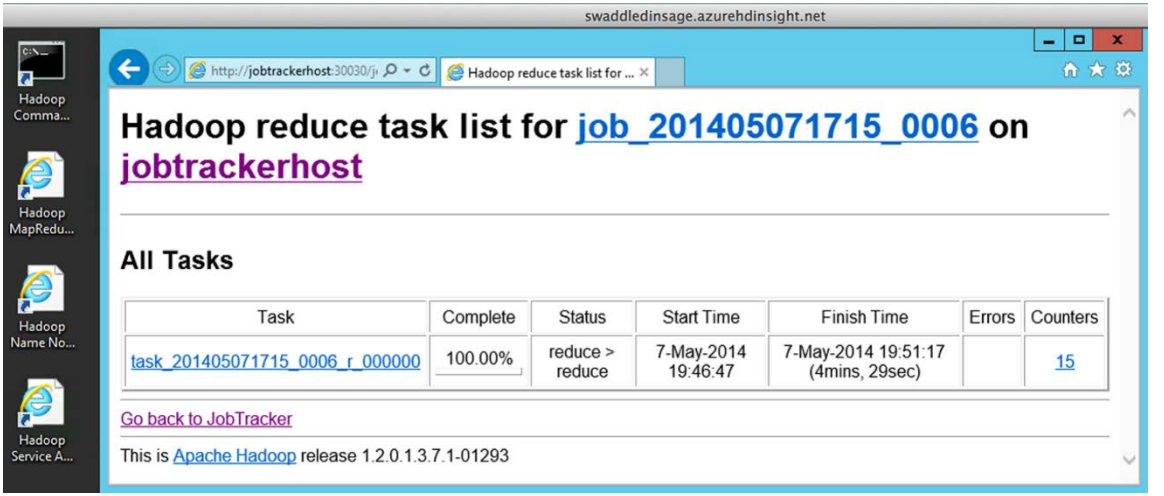


FIGURE 2-13 A list of reduce tasks for a specific MapReduce Job ID job_201405071715_0006.

Oliver continues his review by taking a look at the active TaskTrackers. To do this, he clicks **Nodes** in the Cluster Summary table (see Figure 2-9). Data about the active TaskTracker nodes is show in Figure 2-14.

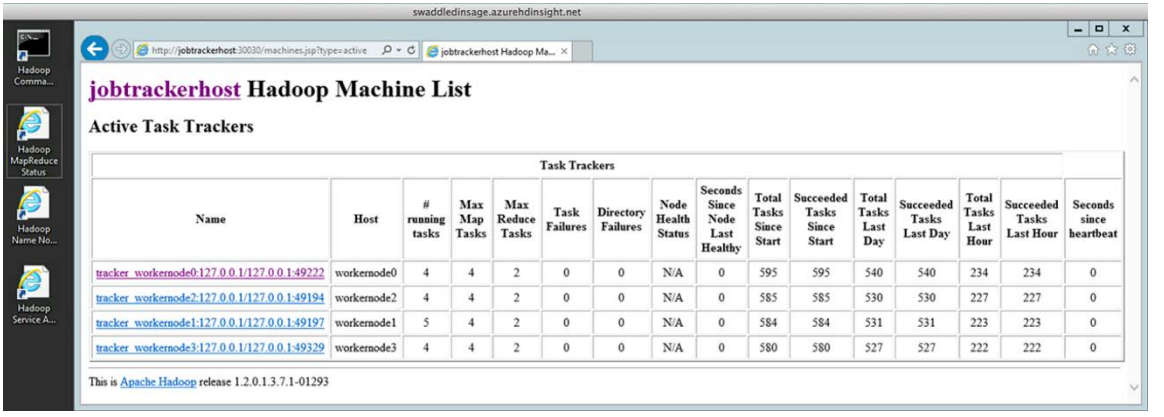


FIGURE 2-14 TaskTracker details for the swaddledinsage HDInsight cluster.

The first thing he sees is a list of all active TaskTrackers and their data. The TaskTracker updates the JobTracker by sending a "heartbeat," letting the JobTracker know it's alive. If a heartbeat is not received, JobTracker will schedule the work on a different TaskTracker. JobTracker also monitors job progress and completion. Oliver notices detailed information about each TaskTracker, such as current and past consumption, along with performance

statistics. He's happy to see that the nodes have had no failures—all tasks have succeeded (between 222 and 234 tasks in the last hour alone).

Scrolling down, Oliver sees the Scheduling Information table. MapReduce jobs are scheduled in Hadoop through the scheduler. Applications such as command-line tools, third-party apps, Ooozie (http://oozie.apache.org/), Hue (http://gethue.com/), and others can use standard Hadoop API calls to submit MapReduce jobs. The scheduler accepts the job by making use of multiple queues, where the jobs wait for resource availability.

Queues can be controlled to provide access to users depending on HDInsight cluster admin requirements. Queues are also used to distribute HDInsight cluster access to multiple applications, which helps Hadoop administrators distribute the resources. Queues can be configured to address requirements such as fast and immediate processing or even resource distributions. Scrolling the Administration webpage, Oliver can see schedule information, which shows the total available schedulers, currently submitted jobs, which jobs are waiting, and more.

The swaddledinsage HDInsight cluster has two schedulers, named *default* and *joblauncher*, and both of the schedulers are ready to accept jobs, as shown in Figure 2-15.

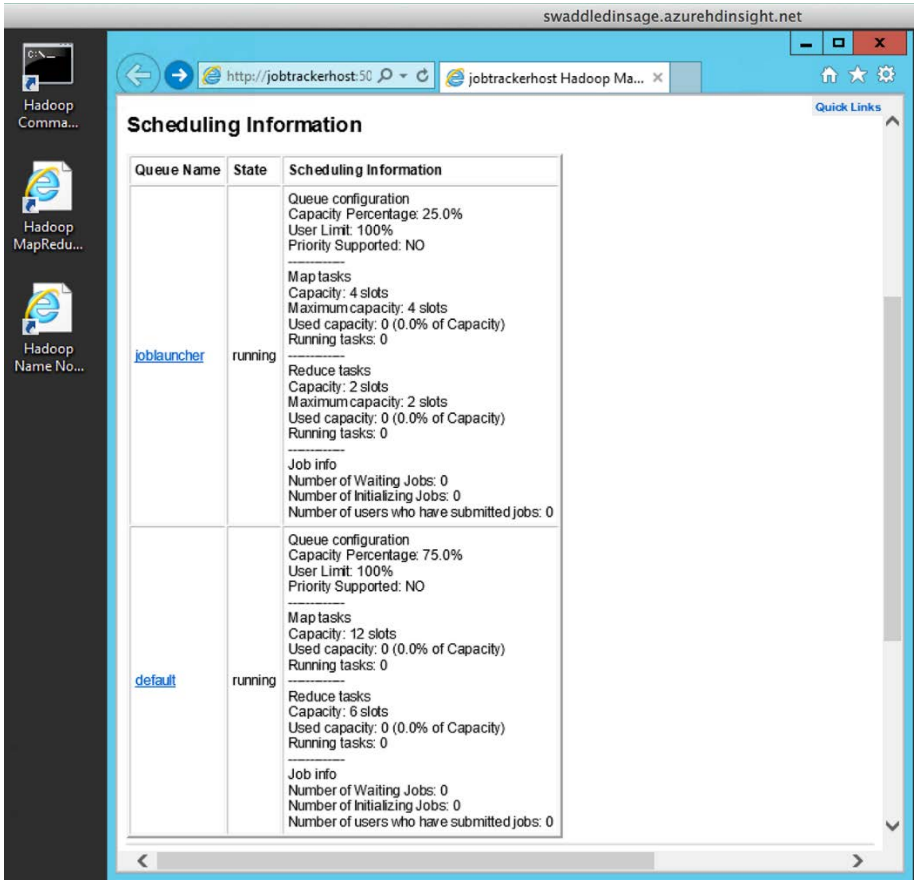


FIGURE 2-15 MapReduce job scheduler information.

Hadoop command line

The Hadoop Command Line shortcut provides command-line access to your HDInsight cluster. You can access HDFS and MapReduce directly from the command line, and you can also use the command line to manage your cluster. Oliver is really eager to try out the Hadoop Command Line shortcut—finally, he can run some jobs.

Several samples are supplied by Microsoft to help new users learn and practice. Oliver decides to use the Hadoop MapReduce pi example. He begins by typing the following command to access the pi sample:

c:\apps\dist\hadoop-1.2.0.1.3.2.0-05>hadoop jar hadoop-examples.jar pi 10 100000

Notice the two parameters in the command: 10 and 100000. Oliver is requesting 10 mappers to calculate the value of pi and display the result value to a .000001 level of precision. The results are shown in Figure 2-16.

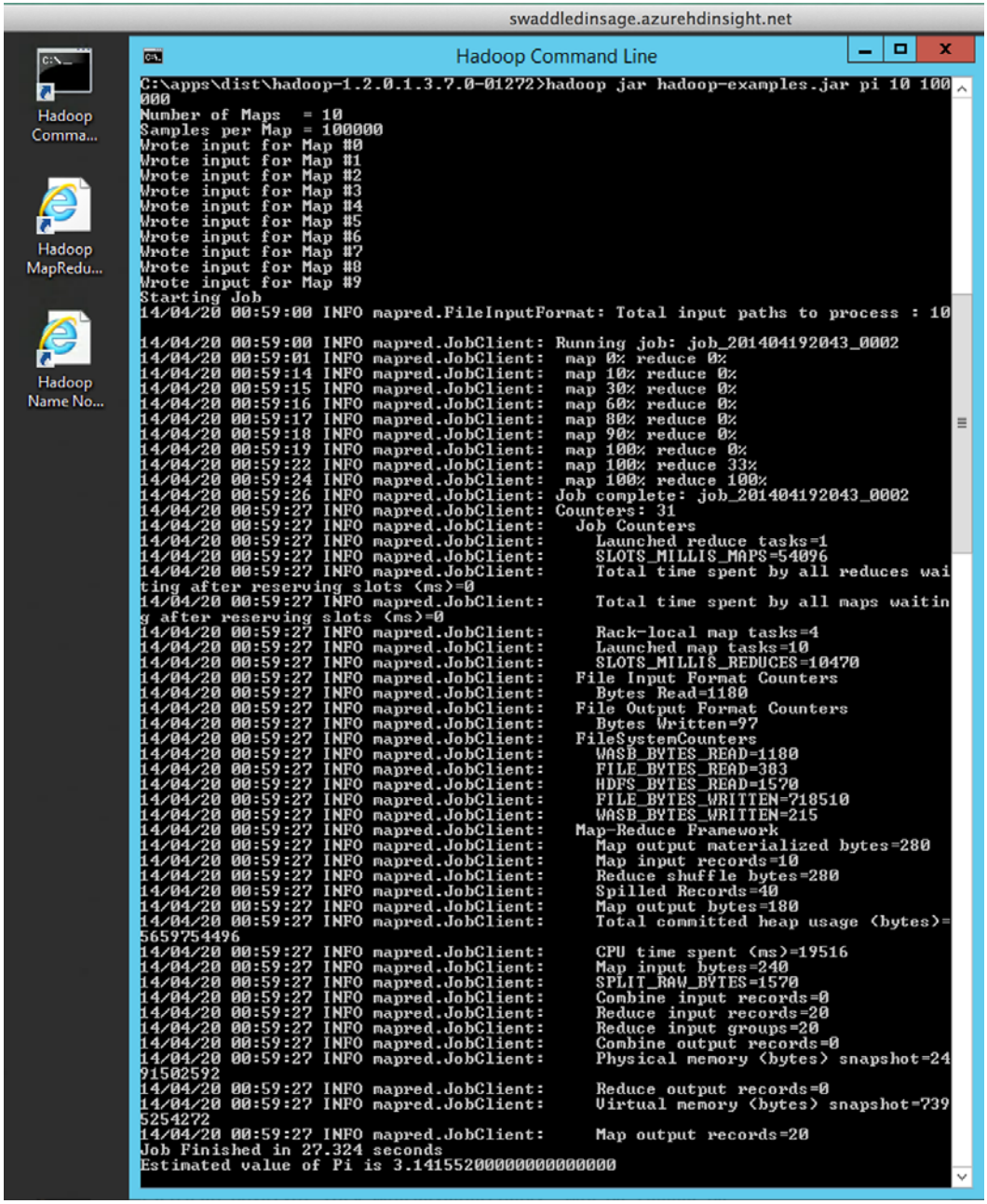


FIGURE 2-16 An example of the results from running the Pi Estimator sample.

Setting up the HDInsight Emulator

The HDInsight Emulator is a dev/test edition of Azure HDInsight that runs locally on Windows desktops and servers. There are three key differences between Azure HDInsight and the HDInsight Emulator:

- Unlike Azure HDInsight, which stores data in Azure Storage, the HDInsight Emulator stores data in your local file system and also processes your data locally with MapReduce.
- The HDInsight Emulator has only a single-node HDInsight cluster, including both the name node and data node in an experimental mode, and the local machine's CPU is used for compute. In contrast, Azure HDInsight supports up to 32 nodes by default and uses Azure Compute.
- HDInsight is a cloud-based service, while the HDInsight Emulator runs on-premises on the following versions of Windows:
    - Windows 7 Service Pack 1
    - Windows Server 2008 R2 Service Pack1
    - Windows 8
    - Windows Server 2012

Now let's explore the HDInsight Emulator a little further.

HDInsight Emulator and Windows PowerShell

The HDInsight Emulator depends on Windows PowerShell, as most of the HDInsight-specific commands are supported through PowerShell. PowerShell gives you UNIX-like command-line access to most of the Windows applications and services. All new Windows operating systems come with native support for PowerShell, so you just need to download the specific PowerShell components for Azure and HDInsight and install them on your machine from the following URL: http://azure.microsoft.com/en-us/documentation/articles/install-configure-powershell/

If you have installed Azure PowerShell 0.7.2 or later, you already have the Azure–supported commands. You can use the link given in the next section to install PowerShell for HDInsight:

Note To learn more about Azure PowerShell, please visit MSDN or the following URL: http://msdn.microsoft.com/en-us/library/windowsazure/jj156055.aspx.

We also provide more details on PowerShell in Chapters 3 and 4.

Installing the HDInsight Emulator

Downloading and installing the HDInsight Emulator is fairly straightforward. Simply follow this URL:

http://www.microsoft.com/web/gallery/install.aspx?appid=HDINSIGHT

HDInsight is installed through the Microsoft Web Platform Installer (WebPI). As you watch the installation's progress, you'll see that the installer downloads and installs Hortonworks Data Platform (HDP) for Windows as part of the HDInsight Emulator. You will also see that Java 1.6, Python 2.7, and a few other third-party components are installed.

After installation is completed, you'll see three shortcuts on your desktop (the same shortcuts we described earlier):

- Hadoop Command Line
- Hadoop Name Node Status
- Hadoop MapReduce Status

In addition, the following folders are created in the disk partition where you installed the HDInsight Emulator:

- Hadoop
- HadoopFeaturePackSetup
- HadoopInstallFiles

Python 2.7 and Java are installed in the following folders:

- Python27:
- Hadoop/Java/

A few other services are also installed with the HDInsight Emulator. If you open the currently configured list of services on your machine (through the Control Panel Services applet), you will see a list of Apache Hadoop–specific services as shown in Figure 2-17. This includes Hadoop services for Hive, Templeton, Oozie, and all the other Hadoop services that ship with HDInsight.

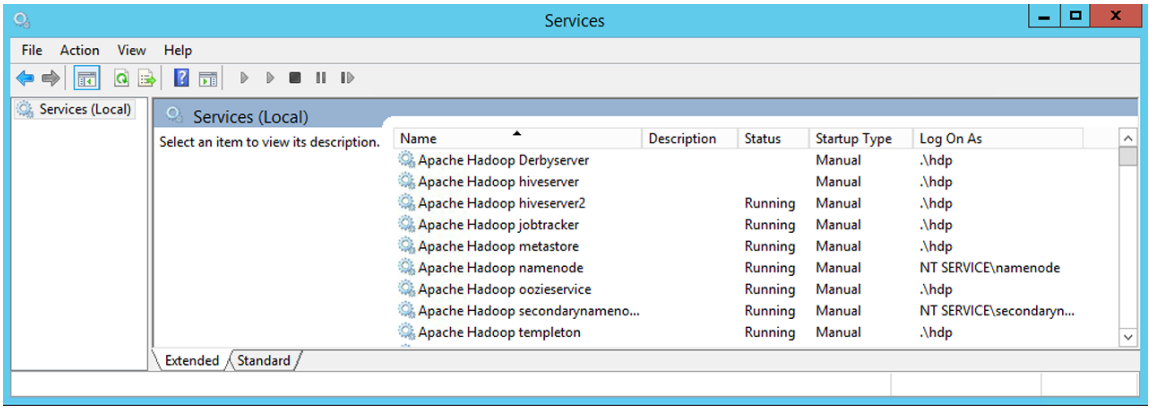


FIGURE 2-17 List of Apache Hadoop-specific services installed with the HDInsight Emulator.

You can learn more about these services from the Hortonworks Data Platform (HDP) for Windows webpage at http://hortonworks.com/products/hdp-windows/.

Using the HDInsight Emulator

Now let's launch the Hadoop Command Line shortcut from the desktop. You'll see a command window as shown in Figure 2-18, along with several Hadoop commands listed. You can type the command `hadoop version` to show the Hadoop version that is installed with the HDInsight Emulator.

Keep in mind that "version" is a class name and that Java is a case-sensitive language. So typing anything other than **version** will return an error.

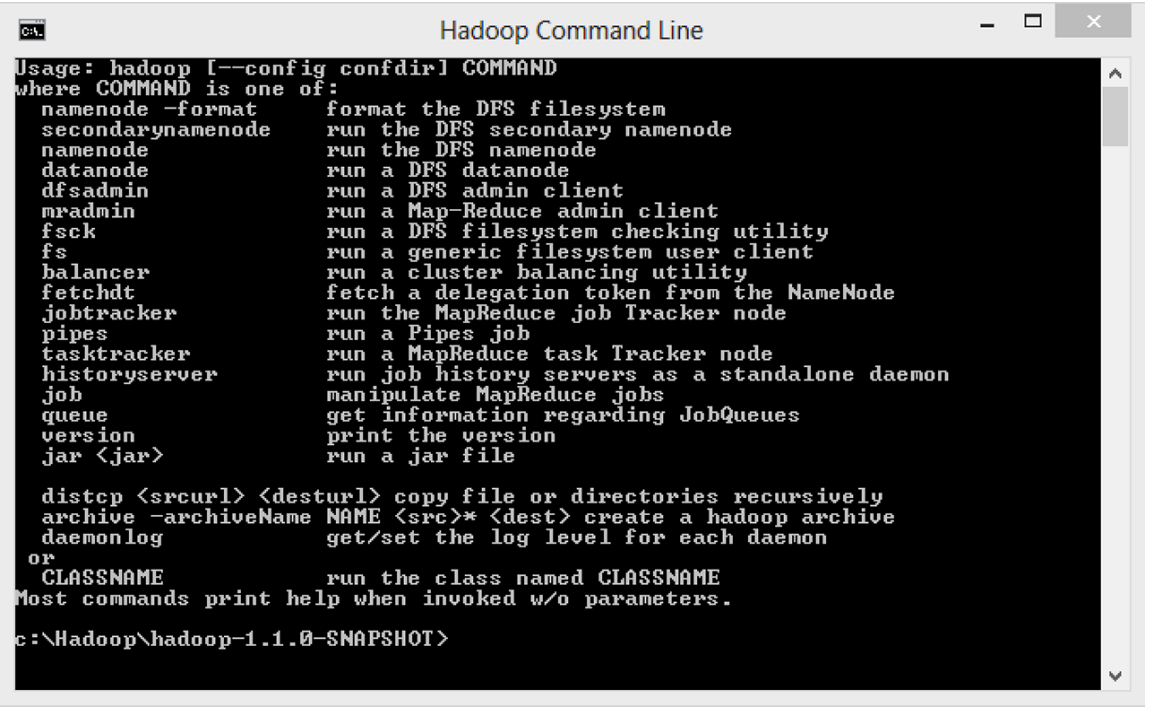


FIGURE 2-18 Running the Hadoop command line shortcut.

As you can see, there are many Hadoop specific commands, and all of these commands can be started by using the format `Hadoop <command_name>`. Some of these command are used for HDFS administration, while others are used to manage the MapReduce service. Use the following commands for HDFS administration:

```
namenode -format, namenode, datanode, dfsadmin, fsck, balancer, fetchdt
```

For MapReduce-specific management and administration, use these commands:

```
Mradmin, job, queue, jobtracker, tasktracker, pipes, historyserver, jar
```

For other HDFS file-specific commands, use **Hadoop fs -<command_name>** -, where most of the commands are Linux file system commands. To see a list of supported Hadoop file system commands, type **Hadoop fs** at the Hadoop command prompt (Figure 2-19).

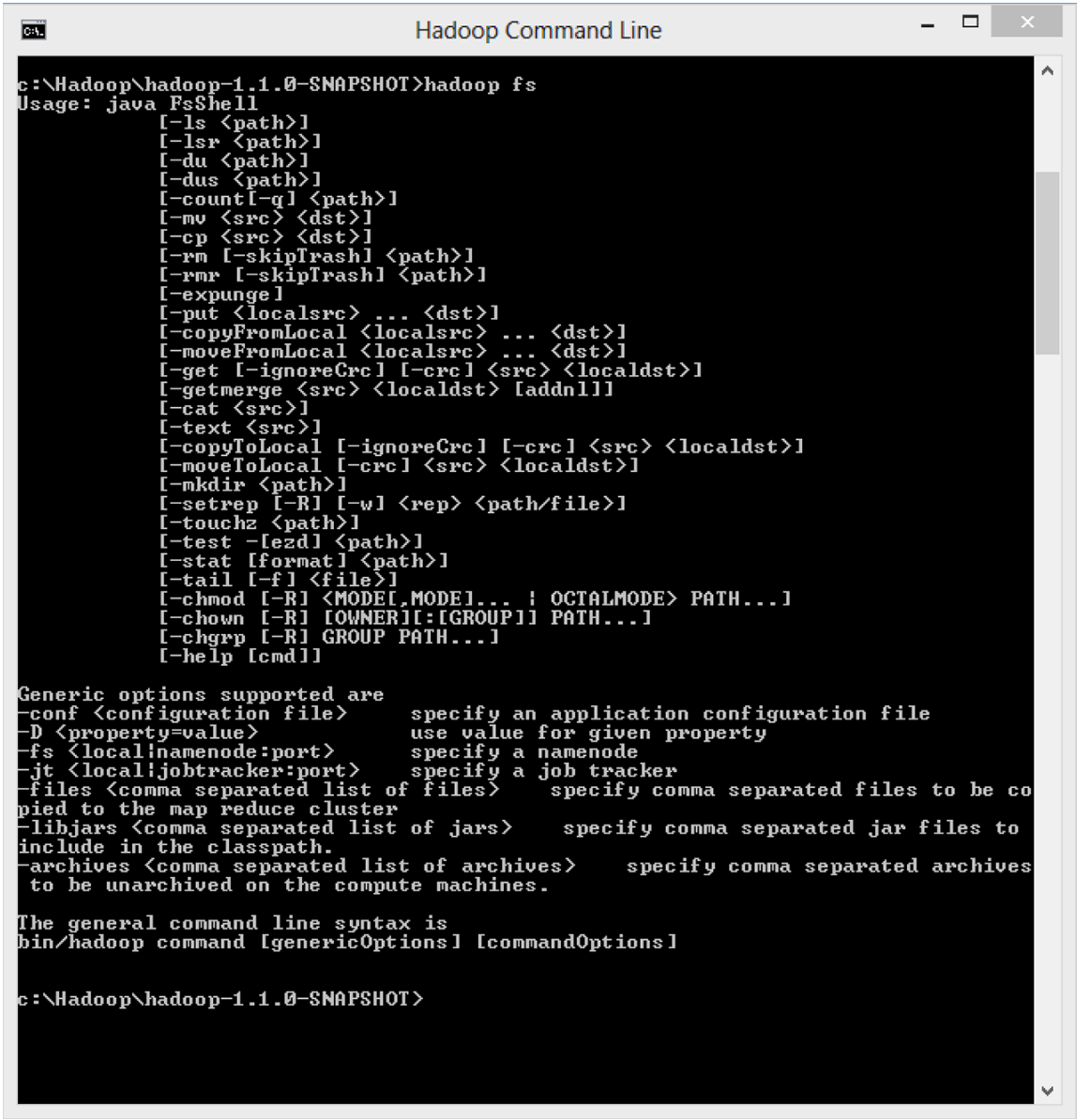


FIGURE 2-19 Supported HDFS commands.

For example, to list the directories and files at the root and also user folders, use the commands `hadoop fs -ls /` and `hadoop fs -ls /user`. Listing 2-1 shows the output

for these commands when we executed them on our demo environment.

LISTING 2-1 Output from the Hadoop command line.

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -ls /

Found 3 items

drwxr-xr-x   - SYSTEM supergroup          0 2013-12-08 22:57 /hive
drwxr-xr-x   - hadoop supergroup          0 2013-12-09 00:04 /mapred
drwxr-xr-x   - SYSTEM supergroup          0 2013-12-08 22:57 /user

c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -ls /user

Found 1 items


drwxr-xr-x   - SYSTEM supergroup          0 2013-12-08 22:57
/user/SYSTEM
```

The Hadoop command line shell can also run other Hadoop applications, which are built on top of the Hadoop framework—applications such as Hive, Pig, HCatalog, Sqoop, Oozie, and many others.

For example, you can run the `pig --help` command (or add `--help` to most other commands) to see the current version for Pig along with help on how to use it. Figure 2-20 displays an example of how you would use Pig commands on the Hadoop command line.

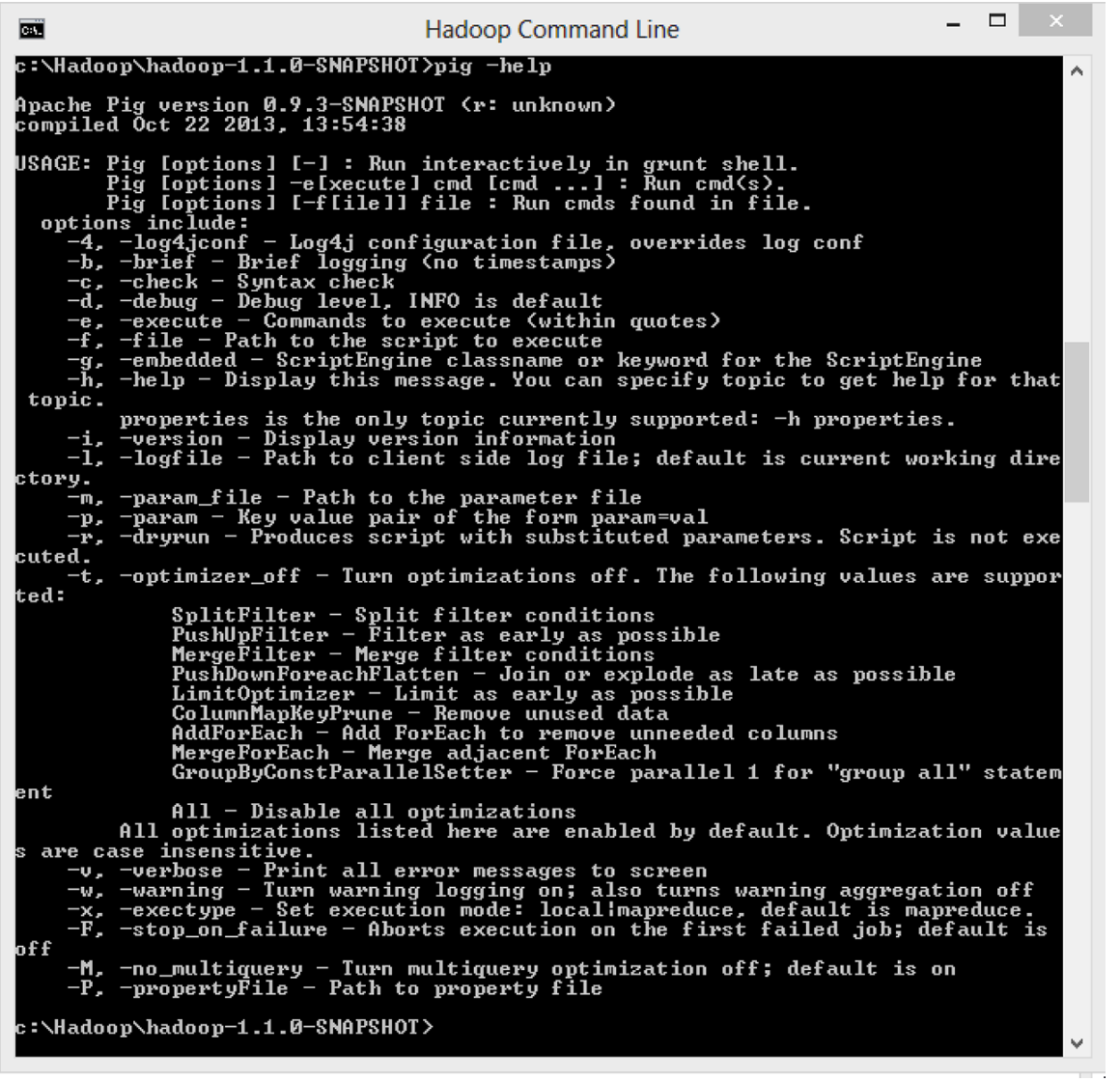


FIGURE 2-20 Using Pig commands on the Hadoop command line.

Name node status

In Hadoop, the name node manages the data stored on all data nodes and also monitors cluster health. As a distributed file system, HDFS stores data on multiple data nodes in

redundant blocks. Hadoop natively provides access to HDFS through the Hadoop web server on port 50070. To see the status of the name node, simply launch your browser and point it to http://Hadoop_host_name:50070.

As mentioned earlier, the Hadoop Name Node Status shortcut is one of the three created on your desktop when you install the HDInsight Emulator. You can see more details about HDFS by clicking this shortcut, which causes the HDInsight Emulator to open the webpage shown in Figure 2-21.

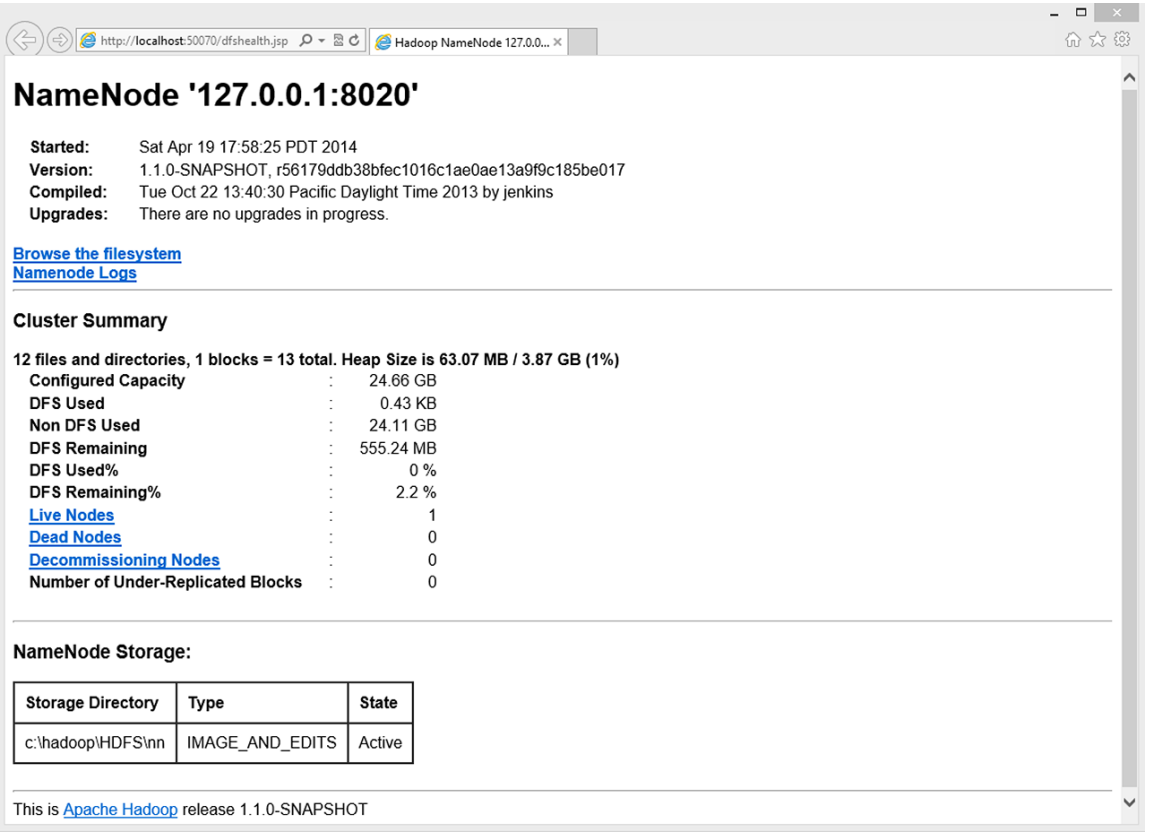


FIGURE 2-21 Hadoop name node status.

The name node status provides information and statistics, including, but not limited to, the following:

- Total number of nodes in the file system

- Total size of the file system, disk space in each node, used and unused space on each disk
- The active data nodes
- Access to HDFS logs

It also provides web access to HDFS if you click **Browse The Filesystem** (Figure 2-22).

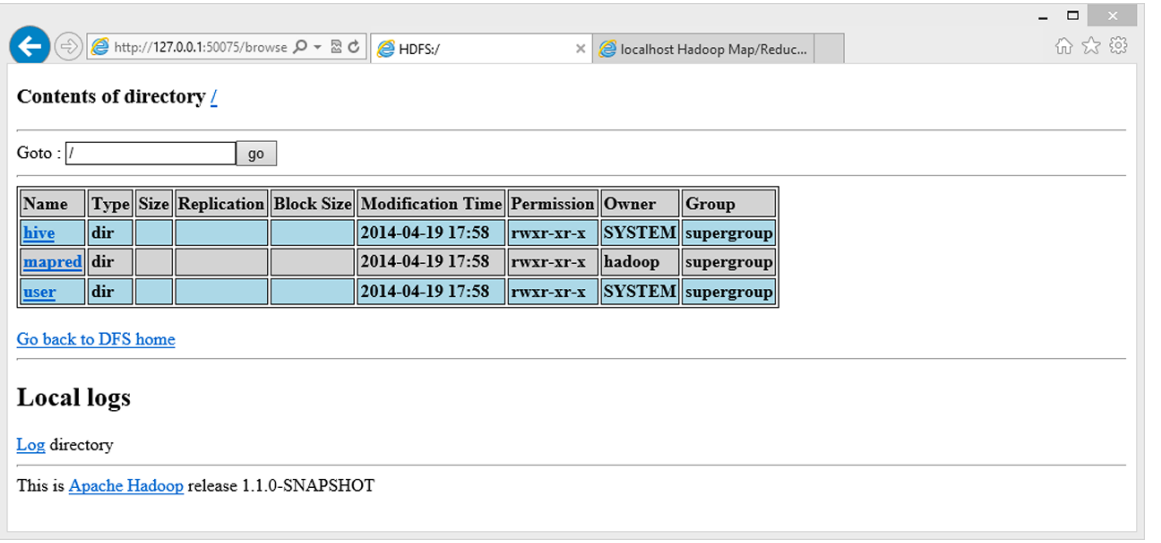


FIGURE 2-22 Web access to HDFS.

MapReduce job status

The Hadoop data-processing layer has two components, JobTracker and TaskTracker. JobTracker usually runs on the same server as the name node. However, if needed, it can also run on a different machine. The role of JobTracker is to accept the jobs submitted to an HDInsight cluster from any source. It uses TaskTracker, which runs on each data node, to complete each job. To see the status of any job running on an HDInsight cluster, use your Hadoop web server on port 50030 or the following URL template: *http://your_hadoop_server:50030/*.

With the HDInsight Emulator you can also view the status of MapReduce jobs by using the Hadoop MapReduce Status shortcut, which opens your browser at port 50030, as shown in Figure 2-23.

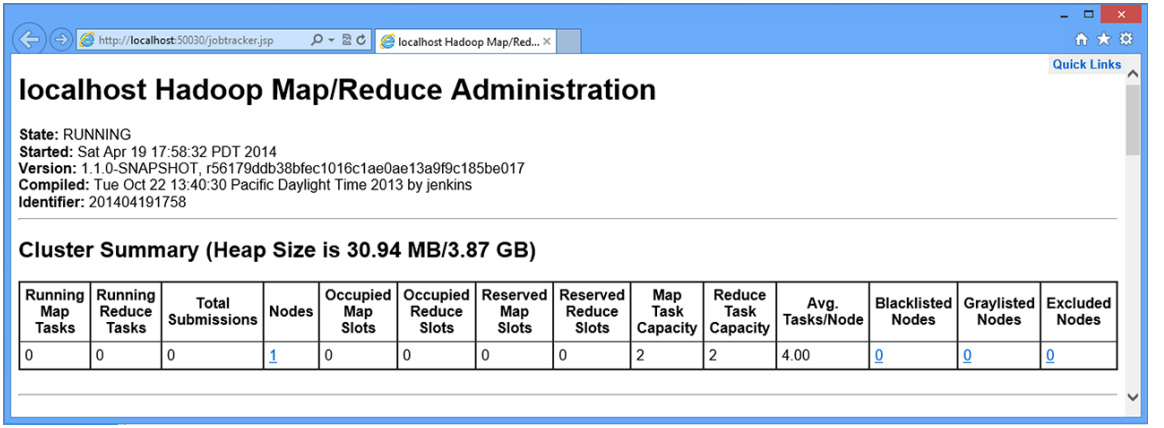


FIGURE 2-23 Name node status in the HDInsight Emulator.

If any jobs are running on the HDInsight cluster, the MapReduce status page provides information about these jobs (as well as previously executed jobs) along with details such as the following:

- Job IDs and other job-specific details for the submitted job
- The status of current and previously run jobs (Completed, Running, Failed)
- Information about total mappers and reducers for the job
- Mapper and reducer counters and statistics such as disk and memory usage

Running the WordCount MapReduce job in the HDInsight Emulator

Oliver is interested in seeing how the HDInsight Emulator works—he foresees his developer team wanting to develop and test jobs on a small subset of data in a local development environment. He'll run a very simple MapReduce job directly on the Hadoop command prompt to understand a little more about Hadoop data storage and processing. In this example, he copies some social media data from the local file system to HDFS and then runs the sample WordCount job on the data to get the count of the most frequently used words in the source data.

The WordCount sample ships with the HDInsight Emulator, so Oliver uses the Hadoop jar command with the parameters shown here to launch the MapReduce job:

```
<Hadoop jar jar_name param1 param2>
```

Create a folder named hdiemulator on the HDFS user directory, and then an input folder under the hdiemulator folder:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hdfs fs -mkdir /user/hdiemulator
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hdfs fs -mkdir /user/hdiemulator/input

c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hdfs fs -ls /user/hdiemulator
Found 1 items
drwxr-xr-x   - Avkash supergroup          0 2013-12-09 00:54 /user/hdiemulator/input
```

Move the text file named sentiment.txt from the local file system to HDFS by using the `Hadoop fs –copyFromLocal` command as shown here:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -copyFromLocal sentiment.txt
/user/hdiemulator/input
```

Verify that the file is correctly located in the input folder by running this command:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -ls /user/hdiemulator/input
Found 1 items
-rw-r--r--   1 oliver supergroup     135891 2013-12-09 01:01
/user/hdiemulator/input/sentiment.txt
```

And ensure that the Hadoop-examples.jar file is available in the local file system:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>dir hadoop-examples.jar
10/22/2013  01:43 PM           146,277 hadoop-examples.jar
```

Once Oliver has verified that the hadoop-examples.jar file is available, he launches a MapReduce job by passing his source file, sentiment.txt. This jar file contains multiple samples, and he'll use the WordCount example defined as a class. The general command to push a MapReduce job to the HDInsight cluster is shown here:

```
Hadoop jar jar_name.jar [class_name, {parameter1, parameter2, parameter2……} ]
```

To start the WordCount sample job on the HDInsight Emulator, Oliver runs this command:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop jar hadoop-examples.jar wordcount
/user/hdiemulator/input /user/hdiemulator/output
```

Oliver is using the source folder on HDFS as /user/hdiemulator/input and specifying the output folder as /user/hdiemulator/output. This is where the output data will be stored when the job is completed.

Once the job is submitted and accepted by the JobTracker, Oliver can see its progress on the output console. The output console lists details such as the job ID and the start of mapper and reducer processes at 0% and 0%, respectively:

```
13/12/09 01:11:40 INFO input.FileInputFormat: Total input paths to process : 1
13/12/09 01:11:41 INFO util.NativeCodeLoader: Loaded the native-hadoop library
13/12/09 01:11:41 WARN snappy.LoadSnappy: Snappy native library not loaded
13/12/09 01:11:41 INFO mapred.JobClient: Running job: job_201312090003_0001
13/12/09 01:11:42 INFO mapred.JobClient:  map 0% reduce 0%
```

At this point Oliver has a couple of options for reviewing the MapReduce job status.

- Open the Hadoop MapReduce Status webpage to display more details about this job.
- Launch another Hadoop command window and use the `Hadoop job` command to get current details about the active MapReduce job.

Once the job is completed, Oliver again has several options for reviewing the MapReduce status and statistics:

- The output console now shows both map and reduce jobs set to 100%. Several other counters show statistics about how disk and memory were used by TaskTrackers in each mapper and reducer, and how much time each mapper and reducer took to complete.

```
13/12/09 01:12:07 INFO mapred.JobClient:  map 100% reduce 100%
13/12/09 01:12:09 INFO mapred.JobClient: Job complete:
job_201312090003_0001
13/12/09 01:12:09 INFO mapred.JobClient: Counters: 29
13/12/09 01:12:09 INFO mapred.JobClient:   Job Counters
…..
13/12/09 01:12:09 INFO mapred.JobClient:     Map output records=6971
```

- To see the job status by using the command line, use the `Hadoop job` command as follows:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop job -status job_201312090003_0001
Job: job_201312090003_0001
file:
hdfs://localhost:8020/hadoop/hdfs/tmp/mapred/staging/Oliver/.staging/job_2
01312090003_0001/job.xml
tracking URL:
http://localhost:50030/jobdetails.jsp?jobid=job_201312090003_0001
map() completion: 1.0
reduce() completion: 1.0
        …..
```

- The MapReduce status webpage displays even more information about the job, as shown in Figure 2-24.

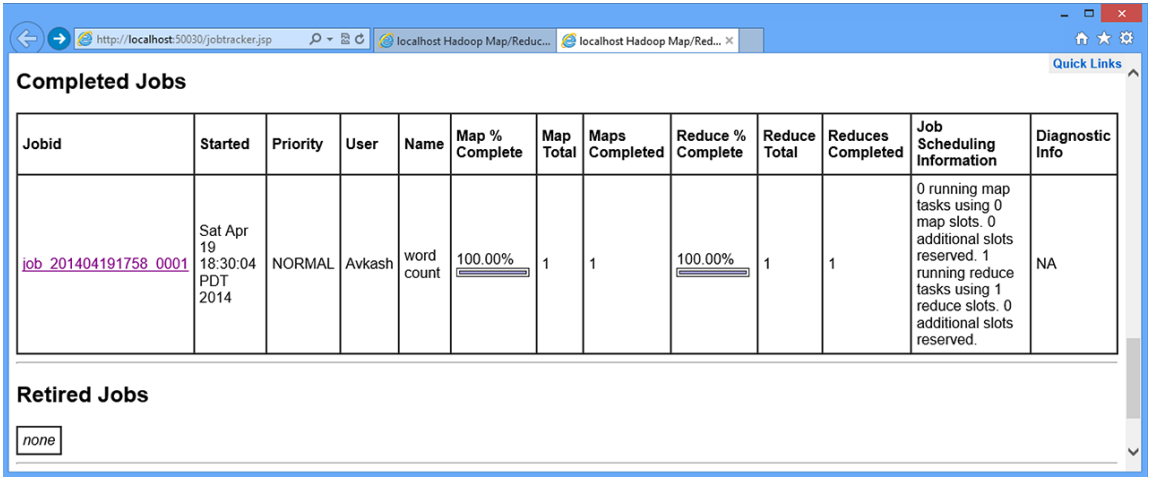


FIGURE 2-24 MapReduce job completion status.

From the web view, Oliver can also search any past job just by using the job ID or other job specific details.

To see the results of the job, he can check the output folder, which is passed as the parameter `/user/hdiemulator/output` with the MapReduce job. Oliver can list the contents of the /user/hdiemulator/output folder by using `hadoop fs –ls`:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -ls /user/hdiemulator/output/
Found 3 items
-rw-r--r--   1 Oliver supergroup          0 2013-12-09 01:12
/user/hdiemulator/output/_SUCCESS
drwxr-xr-x   - Oliver supergroup          0 2013-12-09 01:11
/user/hdiemulator/output/_logs
-rw-r--r--   1 Oliver supergroup      70359 2013-12-09 01:12
/user/hdiemulator/output/part-r-00000
```

Oliver sees the new SUCCESS file, which means the job completed successfully. In MapReduce, once jobs are completed, the results are typically stored in a fixed file named part_r_*NNNNN*, where *NNNNN* is the file counter. With a very large volume of data, the output file counter could grow quite large.

Oliver's result output is stored in file named part_r_00000. He peeks at the file on HDFS by using the `Hadoop fs –tail <filename>` command:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -tail /user/hdiemulator/output/part-r-
00000
```

To copy or move this file from HDFS to his local file system, he can use the command `Hadoop fs –moveToLocal | copyToLocal`:

```
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs -copyToLocal
    /user/hdiemulator/output/part-r-00000 part-r-00000
c:\Hadoop\hadoop-1.1.0-SNAPSHOT>hadoop fs –moveToLocal
    /user/hdiemulator/output/part-r-00000 part-r-00000
```

Summary

In this chapter we introduced HDInsight, a cloud-based Hadoop service on Windows. We also introduced the HDInsight Emulator, which is a single-node experimental HDInsight cluster that runs on-premises. You can provision an HDInsight cluster in Azure using PowerShell or the Azure Management Portal. We also covered how to run MapReduce jobs in the HDInsight cluster from the command prompt.

Chapter 3

Programming HDInsight

With your HDInsight cluster deployed and data loaded, you can begin creating code that turns data into insights. In this chapter, you learn about the various methods of programming with HDInsight, including how to use the HDInsight cmdlets in Windows PowerShell to work with the HDInsight cluster. You'll also learn how to use the HDInsight Software Development Kit (SDK) to develop applications.

Getting started

Our data scientist Oliver has explored HDInsight and run a few MapReduce jobs. He's reported back to the data science team and his manager, Natalie, and told them how quickly he was able to provision a cluster and run MapReduce jobs. Oliver sees a lot of potential for HDInsight, and he and his team are ready for the next step.

Natalie is especially interested in seeing for herself how HDInsight can be used to discover customer insights. Will an investment in HDInsight give *Swaddled in Sage* an advantage over competitors who have yet to invest in a big data solution?

Analytics play an important role in enabling companies to be successful. The ability to efficiently perform analytics on the large volume of data within your organization helps reduce the time required to gain insights. This helps companies stay competitive, as they tap into these insights to guide them in strategic decision making and to enable new online experiences for customers.

As we discussed in Chapter 1, "Big data, quick intro," Azure HDInsight can work with data that is stored in the Hadoop Distributed Files System (HDFS) and in Azure blob storage. To reduce the compute cost that is associated with running an HDInsight cluster, a common approach is to launch the cluster to process the data and then delete the cluster after the data has been processed. By persisting the data in Azure blob storage, you ensure that the data that is processed by HDInsight is not removed after the HDInsight cluster has been deleted. Most importantly, HDInsight provides an HDFS interface on top of Azure blob storage. This interface enables you work with data stored in Azure blob storage with the same tools the Hadoop community uses.

In this chapter, we first show you how to work with the data stored in Azure blob storage and how to programmatically submit and manage MapReduce jobs by using the HDInsight cmdlets in Windows PowerShell. Then we show you how to use the HDInsight .NET SDK to programmatically work with the HDInsight cluster.

> **Note** To run the PowerShell scripts in this chapter, save each of the scripts to a file with the file extension .ps1 and execute them using Windows PowerShell.
>
> Start Windows PowerShell at an elevated command prompt—it needs to run as an administrative account. Then execute the command `Set-ExecutionPolicy RemoteSigned.`
>
> If you have not set up the local environment for using Windows PowerShell and HDInsight, refer to http://azure.microsoft.com/en-us/documentation/articles/hdinsight-get-started/.

> **Tip** See http://msdn.microsoft.com/en-us/library/dn479228.aspx for the reference documentation for the Azure HDInsight cmdlets.

MapReduce jobs and Windows PowerShell

In Chapter 2, "Getting started with HDInsight," Oliver ran the sample WordCount MapReduce job in the HDInsight Emulator. In this exercise, Oliver will programmatically submit the same WordCount MapReduce job using Windows PowerShell. He'll work with the swaddledinsage HDInsight cluster and use the PowerShell cmdlets to start the job. Unlike in the Emulator job, the WordCount MapReduce program is available in the hadoop-examples.jar file that has already been deployed to the Azure blob storage used by HDInsight. Oliver also has a data file name sentiment.txt, which includes details about customer sentiment from a recent *Swaddled in Sage* survey. Oliver has already uploaded the sentiment.txt file to an Azure blob storage account named sagestorage, which he created in Chapter 2 as well. The sentiment.txt file can be seen in the Azure Storage container shown in Figure 3-1, directly from the Microsoft Azure Management Portal.

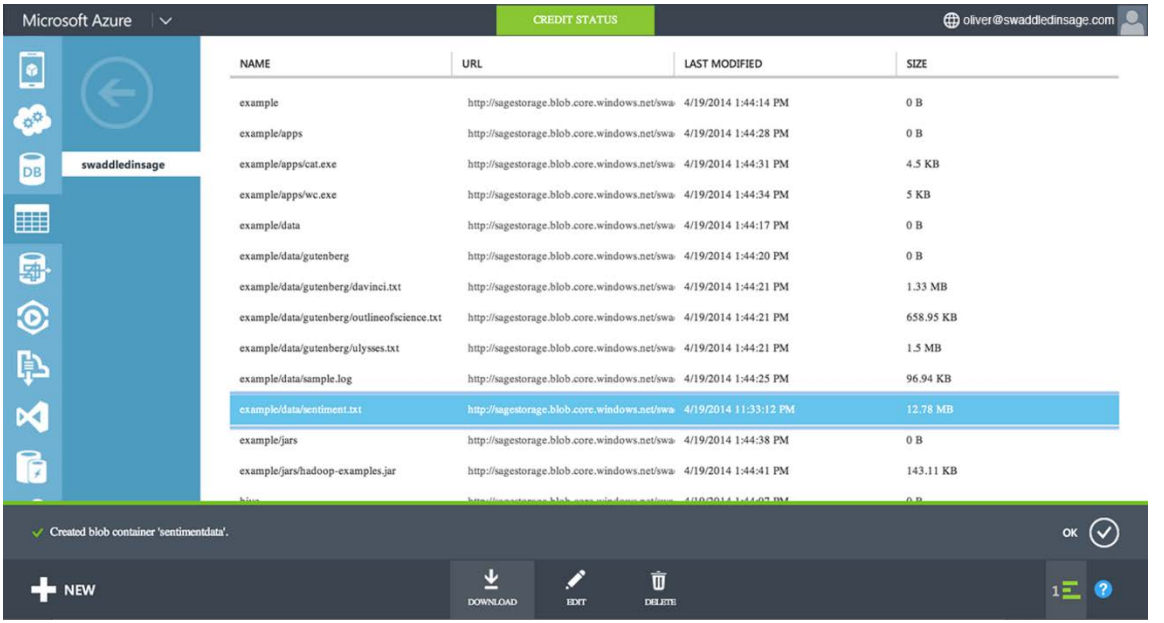


FIGURE 3-1 Azure Storage container showing the data source sentiment.txt.

Oliver will need to work with the following cmdlets:

- `New-AzureHDInsightMapReduceJobDefinition` Creates the job definition corresponding to the MapReduce job.
- `Start-AzureHDInsightJob` Starts the job on the cluster.
- `Wait-AzureHDInsightJob` Waits for the job to complete instead of exiting or polling the cluster for the status of the job.
- `Get-AzureHDInsightJobOutput` Displays the output.

The cmdlet `Start-AzureHDInsightJob` starts the job on the HDInsight cluster. When the MapReduce job is completed, it writes the results to the output location specified in the job definition output folder; Oliver is using the folder sentimentCountOutput as the output folder. Listing 3-1 shows the script Oliver uses.

LISTING 3-1 Script for working with an HDInsight cluster.

```
# Step 1 - Specify the name of the Microsoft Azure subscription, and
# the name of the Microsoft Azure Storage account
$subscriptionName = "BizSpark"
$clusterName = "swaddledinsage"
```

```
# Step 2 - Create the MapReduce job
$wcJobDefinition = New-AzureHDInsightMapReduceJobDefinition -JarFile
"wasb:///example/jars/hadoop-examples.jar" -ClassName "wordcount" -Arguments
"wasb:///example/data/sentiment.txt", "wasb:///example/data/sentimentCountOutput"

# Step 3 - Start the HDInsight job
$wcJob = Start-AzureHDInsightJob -Cluster $clusterName -Subscription $subscriptionName
-JobDefinition $wcJobDefinition

# Step 4 - Wait for job completion
Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $wcJob -
WaitTimeoutInSeconds 3600

# Step 5 - Show the output of the job
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $wcJob.JobId -StandardOutput
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $wcJob.JobId -StandardError
```

After the job completes execution, Oliver sees the output shown in Listing 3-2 because he used the `Get-AzureHDInsightJobOutput` cmdlet.

LISTING 3-2 Output from the HDInsight job.

```
StatusDirectory : 1653c797-1578-4aee-91bd-89cc874c9594
ExitCode        : 0
Name            : wordcount
Query           :
State           : Completed
SubmissionTime  : 04/19/2014 20:12:10
Cluster         : sage
PercentComplete : map 100% reduce 100%
JobId           : job_201404191758_0012

14/04/19 20:12:12 INFO input.FileInputFormat: Total input paths to process : 1
14/04/19 20:12:12 WARN snappy.LoadSnappy: Snappy native library is available
14/04/19 20:12:12 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/04/19 20:12:12 INFO snappy.LoadSnappy: Snappy native library loaded
14/04/19 20:12:12 INFO mapred.JobClient: Running job: job_201404191758_0012
14/04/19 20:12:23 INFO mapred.JobClient:  map 0% reduce 0%
14/04/19 20:12:25 INFO mapred.JobClient:  map 100% reduce 0%
14/04/19 20:12:28 INFO mapred.JobClient:  map 100% reduce 33%
14/04/19 20:12:34 INFO mapred.JobClient:  map 100% reduce 100%
14/04/19 20:12:34 INFO mapred.JobClient: Job complete: job_201404191758_0012
14/04/19 20:12:34 INFO mapred.JobClient: Counters: 29
14/04/19 20:12:34 INFO mapred.JobClient:   Job Counters
14/04/19 20:12:34 INFO mapred.JobClient:     Launched reduce tasks=1
14/04/19 20:12:34 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=8733
14/04/19 20:12:34 INFO mapred.JobClient:     Total time spent by all reduces waiting
```

```
after reserving slots
    (ms)=0
14/04/19 20:12:34 INFO mapred.JobClient:     Total time spent by all maps waiting
after reserving slots
    (ms)=0
14/04/19 20:12:34 INFO mapred.JobClient:     Launched map tasks=1
14/04/19 20:12:34 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=10516
14/04/19 20:12:34 INFO mapred.JobClient:   File Output Format Counters
14/04/19 20:12:34 INFO mapred.JobClient:     Bytes Written=337623
14/04/19 20:12:34 INFO mapred.JobClient:   FileSystemCounters
14/04/19 20:12:34 INFO mapred.JobClient:     WASB_BYTES_READ=1395666
14/04/19 20:12:34 INFO mapred.JobClient:     FILE_BYTES_READ=466915
14/04/19 20:12:34 INFO mapred.JobClient:     HDFS_BYTES_READ=163
14/04/19 20:12:34 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=1053915
14/04/19 20:12:34 INFO mapred.JobClient:     WASB_BYTES_WRITTEN=337623
14/04/19 20:12:34 INFO mapred.JobClient:   File Input Format Counters
14/04/19 20:12:34 INFO mapred.JobClient:     Bytes Read=1395667
14/04/19 20:12:34 INFO mapred.JobClient:   Map-Reduce Framework
14/04/19 20:12:34 INFO mapred.JobClient:     Map output materialized bytes=466761
14/04/19 20:12:34 INFO mapred.JobClient:     Map input records=32118
14/04/19 20:12:34 INFO mapred.JobClient:     Reduce shuffle bytes=466761
14/04/19 20:12:34 INFO mapred.JobClient:     Spilled Records=65912
14/04/19 20:12:34 INFO mapred.JobClient:     Map output bytes=2387798
14/04/19 20:12:34 INFO mapred.JobClient:     Total committed heap usage
(bytes)=1029046272
14/04/19 20:12:34 INFO mapred.JobClient:     CPU time spent (ms)=7498
14/04/19 20:12:34 INFO mapred.JobClient:     Combine input records=251357
14/04/19 20:12:34 INFO mapred.JobClient:     SPLIT_RAW_BYTES=163
14/04/19 20:12:34 INFO mapred.JobClient:     Reduce input records=32956
14/04/19 20:12:34 INFO mapred.JobClient:     Reduce input groups=32956
14/04/19 20:12:34 INFO mapred.JobClient:     Combine output records=32956
14/04/19 20:12:34 INFO mapred.JobClient:     Physical memory (bytes)
snapshot=494903296
14/04/19 20:12:34 INFO mapred.JobClient:     Reduce output records=32956
14/04/19 20:12:34 INFO mapred.JobClient:     Virtual memory (bytes)
snapshot=1430323200
14/04/19 20:12:34 INFO mapred.JobClient:     Map output records=251357
```

To see the job output, shown in Listing 3-3, Oliver opens the Azure Storage Explorer (or any Azure Storage browsing tool) and navigates to the sentimentCountOutput directory. The output file name is part-r-00000; the file is inside the sentimentCountOutput directory. Oliver can also download the MapReduce job's result file from Azure Storage to his local machine by doing one of the following:

- Running the `Get-AzureStorageBlobContent` cmdlet.
- Downloading the file by using third-party tools for Azure Storage.

- Downloading the file from Azure blob storage directly from the Azure Management Portal, as shown in Figure 3-2.

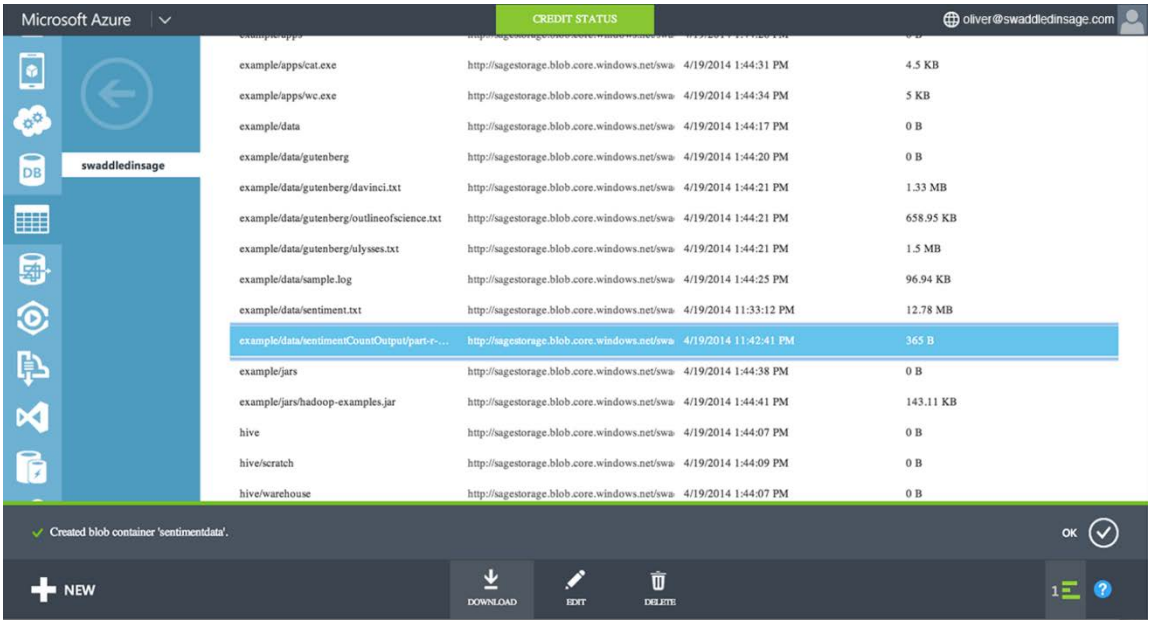


FIGURE 3-2 Downloading job results from a MapReduce job from the Azure Management Portal.

When Oliver reviews the output, several things catch his attention. The terms *return*, *shipping*, and *policy* are high on the list—he'll ask Peter to take a look at *Sage*'s return policy. In an age where brief is best, it's interesting to see that *swaddledinsage* ranks lower than its abbreviation, *SIS*, and *love* ranks lower than *luv*. He's not surprised to see *spring* on the list because *Swaddled In Sage* is currently previewing its spring line. And the appearance of *children* on the list may hint at an area of expansion for the company; it currently does not sell children's clothing.

> **Note** This is only a subset of the output and displays only those words within the top 15 count.

LISTING 3-3 Job results output.

```
cool            86462
children        18785
hate            20349
love            6917
luv             11058
```

```
policy              20961
return              121958
sage                1614215
shipping            86445
sis                 19499
size                21590
spring              33440
swaddledinsage      9771
trendy              39083
ugly                20349
```

Hadoop streaming

As you explore HDInsight, you will notice that all the sample programs are written using Java. This is because the core Hadoop codebase is written in Java.

But can you write a non-Java MapReduce program? And how does a non-Java MapReduce program work with data that is stored in HDFS or Azure Storage? In this section, we explain how to quickly get started writing MapReduce programs based on Hadoop streaming and how to submit the non-Java HDInsight job to the cluster for execution.

Hadoop streaming provides the ability to execute programs that have mapper or reducer functionalities developed in any programming language. When these programs execute in an HDInsight cluster, separate processes are created for each mapper and reducer. These processes are just regular processes that you can see running on the Windows operating system. Hadoop streaming programs take care of passing the input data to each mapper and reducer via the input of the process (referred to as `stdin`). When the mapper and reducer programs write to the output (referred to as the `stdout`), each line is converted into a key/value pair and written back to storage. The key and value are separated with a tab character ('\t') as a delimiter. The delimiter can be changed if you like (some options are shown later).

When you provision an HDInsight cluster, the Java Runtime (JAR) file hadoop-streaming.jar, which provides the Hadoop streaming capabilities, is automatically added to the Azure Storage used by the HDInsight cluster. To get started with Hadoop streaming, the first thing you do is write the mapper and reducer programs.

In this exercise, Oliver uses C# to write the mapper and reducer programs and then copy the executables to the storage used by the sage cluster. He then runs the HDInsight cmdlet

`New-AzureHDInsightStreamingMapReduceJobDefinition` to define an HDInsight job that uses Hadoop streaming and the cmdlet `Start-AzureHDInsightJob` to start the job on the HDInsight cluster.

Write a Hadoop streaming mapper and reducer using C#

Using Hadoop streaming, both the mapper and reducer programs read their respective inputs from `stdin`. The output is sent to `stdout`. By default, we assume that the data for `stdin` and `stdout` is written line by line. If the input to `stdin` is not line-by-line data (e.g., binary data), you need to specify the input format being used when you invoke Hadoop streaming by using the `-inputformat` parameter. When each of the mappers and reducers is started, the corresponding executable is created as a separate process.

> **Tip** Use the executables that are included in the HDInsight storage account to get started. Use /example/apps/cat.exe and /example/apps/wc.exe as the mapper and reducer programs, respectively.

Oliver writes two console applications in C#, providing both mapper and reducer functionality in separate classes. The mapper console program shown in Listing 3-4 reads each line from `stdin` (unless otherwise specified) and splits the line into a collection of words based on the delimiters specified (`' ', ',', '.', ':', '\t', '<', '/', '>'`). The code flow is very simple here: for each word in the collection, the `Mapper` class outputs the word and the value `1` to `stdout`.

LISTING 3-4 Mapper console program.

```
using System;
using System.IO;

namespace MapReduceExample {
    public class Mapper {
        static void Main(string[] args)
        {
            String line;

            if (args.Length > 0)
                Console.SetIn(new StreamReader(args[0]));
            while ((line = Console.ReadLine()) != null) {
                    string[] split = line.Split(new Char[] { ' ', ',', '.',
                    ':', '\t', '<', '/', '>' });

                    foreach (string word in split)
                    Console.WriteLine(word + "\t1");
```

```
            }
        }
    }
}
```

Oliver finishes his MapReduce program by writing reducer functionality in the `Reduce` class as another console application (Listing 3-5). Similar to the mapper program, the reducer program reads from `stdin`. Hadoop streaming takes care of reading from the output produced by the mapper program and sends the results to each reducer program. Most importantly, Hadoop streaming ensures that the intermediate results are sorted before the words are sent to each reducer. Hence, all the entries that correspond to a word are sent to a single reducer.

LISTING 3-5 Reducer console program.

```
using System;
using System.IO;
using System.Collections;

namespace MapReduceExample {
    public class Reducer
    {
        static void Main(string[] args)
        {
            String line;
            Hashtable wordHash = new Hashtable();

        if (args.Length > 0)
            Console.SetIn(new StreamReader(args[0]));

        while ((line = Console.ReadLine()) != null) {
            string[] split = line.Split(new Char[] { '\t' });
                string word = split[0];

                if (wordHash.ContainsKey(word))
                {
                    int count = (int)wordHash[word];
                    wordHash[word] = (int)wordHash[word] + 1;
                }
                else
                    wordHash[word] = 1;

            }

            foreach (string key in wordHash.Keys)
            {
                Console.WriteLine(key + "\t" + (int)wordHash[key]);
            }
```

```
        }
    }
}
```

Now that Oliver has successfully composed the C# programs, two executables files are produced: Mapper.exe and Reducer.exe. These files need to be in the sagestorage account that is attached to the swaddledinsage HDInsight cluster in order for HDInsight to use them. In the next exercise, Oliver writes another HDInsight streaming job to upload these files to the Azure Storage account that is used by the HDInsight cluster.

Run the HDInsight streaming job

As we mentioned, the C# programs Oliver just wrote produced two executable files: Mapper.exe and Reducer.exe. To upload these files to the sagestorage Azure Storage account, he can use various third-party tools or Visual Studio 2013. The following steps are taken to upload the files using Visual Studio 2013:

1. Oliver selects the sagestorage HDInsight storage account. If this is the first time he is using Visual Studio 2013 Server Explorer to add a new Azure Storage account, he needs to click the Storage node, right-click, and choose **Attach External Storage**.

2. Once the swaddledinsage container is selected from sagestorage blob storage, Oliver clicks the **Upload** icon and selects both executable files (Mapper.exe and Reducer.exe) from the local machine to upload. Figure 3-3 shows the uploaded executable files in Azure Storage.

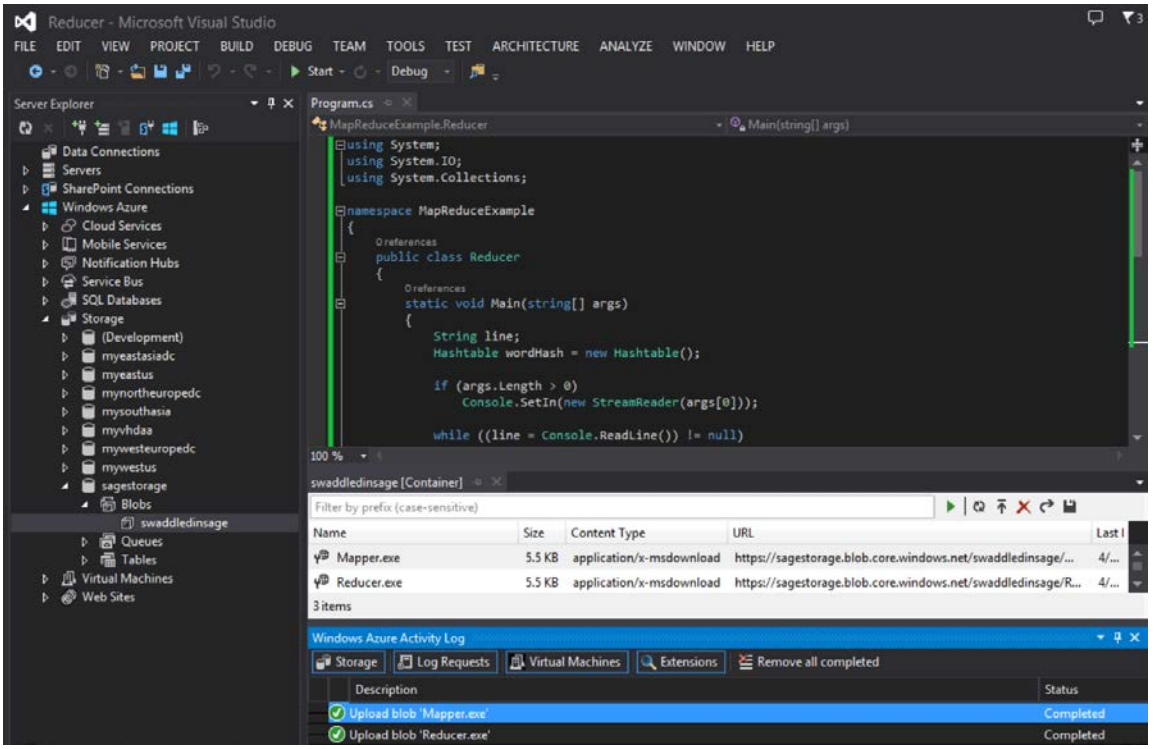


FIGURE 3-3 The mapper and reducer programs in Azure Storage.

Next, Oliver uses the PowerShell script shown in Listing 3-6 to create and start the MapReduce job on the sage cluster. In step 2 of the script, the following switches are used by the `New-AzureHDInsightStreamingMapReduceJobDefinition` cmdlet:

- `-Files` Specifies the location of the files that are used in the streaming MapReduce job. In this example, two files are specified: /example/apps/wc.exe and /example/apps/wc.exe.
- `-Mapper` Specifies the mapper program used in the streaming MapReduce job.
- `-Reducer` Specifies the reducer program used in the streaming MapReduce job.

- `-InputPath` Specifies the location of the input data.
- `-OutputPath` Specifies the location where the output of the mapper and reducer programs will be written.

LISTING 3-6 Running the Hadoop streaming job.

```
# Step 1 - Specify the name of the Microsoft Azure subscription, and
# the name of the Microsoft Azure Storage account
$subscriptionName = "BizSpark"
$clusterName = "swaddledinsage"

# Step 2 - Create the MapReduce job for the Hadoop streaming job
$streamingWCJobDefinition = New-AzureHDInsightStreamingMapReduceJobDefinition  -Files
"Mapper.exe",  "Reducer.exe"  -InputPath "/example/data/sentiment.txt"  -OutputPath
"/example/data/StreamingOutput/wc"  -Mapper "Mapper.exe" -Reducer "Reducer.exe"

# Step 3 - Start the HDInsight job
$wcJob = Start-AzureHDInsightJob -Cluster $clusterName -Subscription $subscriptionName
-JobDefinition $streamingWCJobDefinition

# Step 4 - Wait for  job completion
Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $wcJob -
WaitTimeoutInSeconds 3600

# Step 5 - Show the output of the job
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $wcJob.JobId -StandardOutput
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $wcJob.JobId -StandardError
```

The output is displayed on the screen and the results, shown in Listing 3-7, are written to a file.

LISTING 3-7 Results of running the Hadoop streaming job.

```
StatusDirectory : 41da357f-e62a-4450-96cc-5330b8979a3d
ExitCode        : 0
Name            : Mapper.exe
Query           :
State           : Completed
SubmissionTime  : 04/20/2014 00:41:10
Cluster         : swaddledinsage
PercentComplete : map 100%  reduce 100%
JobId           : job_201404191758_0024

packageJobJar: [] [/C:/apps/dist/hadoop-1.2.0.1.3.7.0-01272/lib/hadoop-streaming.jar]
D:\Users\hdp\AppData\Local\Temp\streamjob364756251437869647.jar tmpDir=null

14/04/20 00:41:10 WARN snappy.LoadSnappy: Snappy native library is available
```

```
14/04/20 00:41:10 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/04/20 00:41:10 INFO snappy.LoadSnappy: Snappy native library loaded
14/04/20 00:41:12 INFO mapred.FileInputFormat: Total input paths to process : 1
14/04/20 00:41:17 INFO streaming.StreamJob: getLocalDirs(): [c:\hdfs\mapred\local]
14/04/20 00:41:17 INFO streaming.StreamJob: Running job: job_201404191758_0024
14/04/20 00:41:17 INFO streaming.StreamJob: To kill this job, run:
14/04/20 00:41:18 INFO streaming.StreamJob: C:\apps\dist\hadoop-1.2.0.1.3.7.0-
01272/bin/hadoop job
    -Dmapred.job.tracker=jobtrackerhost:9010 -kill job_201404191758_0024
14/04/20 00:41:20 INFO streaming.StreamJob: Tracking URL:
    http://headnodehost:50030/jobdetails.jsp?jobid= job_201404191758_0024
14/04/20 00:41:21 INFO streaming.StreamJob:  map 0%  reduce 0%
14/04/20 00:41:26 INFO streaming.StreamJob:  map 50%  reduce 0%
14/04/20 00:41:30 INFO streaming.StreamJob:  map 100%  reduce 0%
14/04/20 00:41:32 INFO streaming.StreamJob:  map 100%  reduce 33%
14/04/20 00:41:36 INFO streaming.StreamJob:  map 100%  reduce 100%
14/04/20 00:41:40 INFO streaming.StreamJob: Job complete: job_201404191758_0024
14/04/20 00:41:41 INFO streaming.StreamJob: Output: /example/data/StreamingOutput/wc
```

To see the results, Oliver navigates to /example/data/StreamingOutput/wc. The file containing the results of our job—the frequency of words that appeared in the file /example/data/sentiment.txt—is saved to a file named part-r-00000. Oliver downloads the file part-r-00000 from Azure blob storage to his local machine. He then uses the PowerShell cmdlet `Get-AzureStorageBlobContent` to download the part-r-00000 file from sagestorage blob storage:

```
$storageAccountName = "sagestorage"
$ContainerName = "swaddledinsage"

$storageAccountKey = Get-AzureStorageKey $storageAccountName | %{ $_.Primary }
$storageContext = New-AzureStorageContext –StorageAccountName $storageAccountName –
StorageAccountKey $storageAccountKey

Get-AzureStorageBlobContent -Container $ContainerName -Blob
example/data/StreamingOutput/wc/part-00000 -Context $storageContext –Force
```

The results downloaded are similar to what Oliver saw earlier after running the WordCount program that was provided as part of examples/jars/hadoop-examples.jar.

Using the HDInsight .NET SDK

So far, Oliver has been using PowerShell to interact with his swaddledinsage HDInsight cluster. HDInsight also provides the HDInsight .NET SDK, which has APIs for interacting with a cluster, managing a cluster, and handling job submissions. Oliver decides to give the HDInsight .NET SDK a try to learn more about how to interact with his HDInsight cluster .

The SDK is available as a NuGet package and can be installed using the NuGet Library Package Manager in Visual Studio.

> **Tip** NuGet is the package manager for the Microsoft Development platform. It provides developers with the ability to download and use .NET libraries within their applications. To learn more about NuGet, see http://www.nuget.org/.

Oliver chooses Visual Studio to create a new C# console application project. After the project is created, he starts the Package Manager Console (Tools, NuGet Package Manager, Package Manager Console). Figure 3-4 shows how to launch the Package Manager Console.

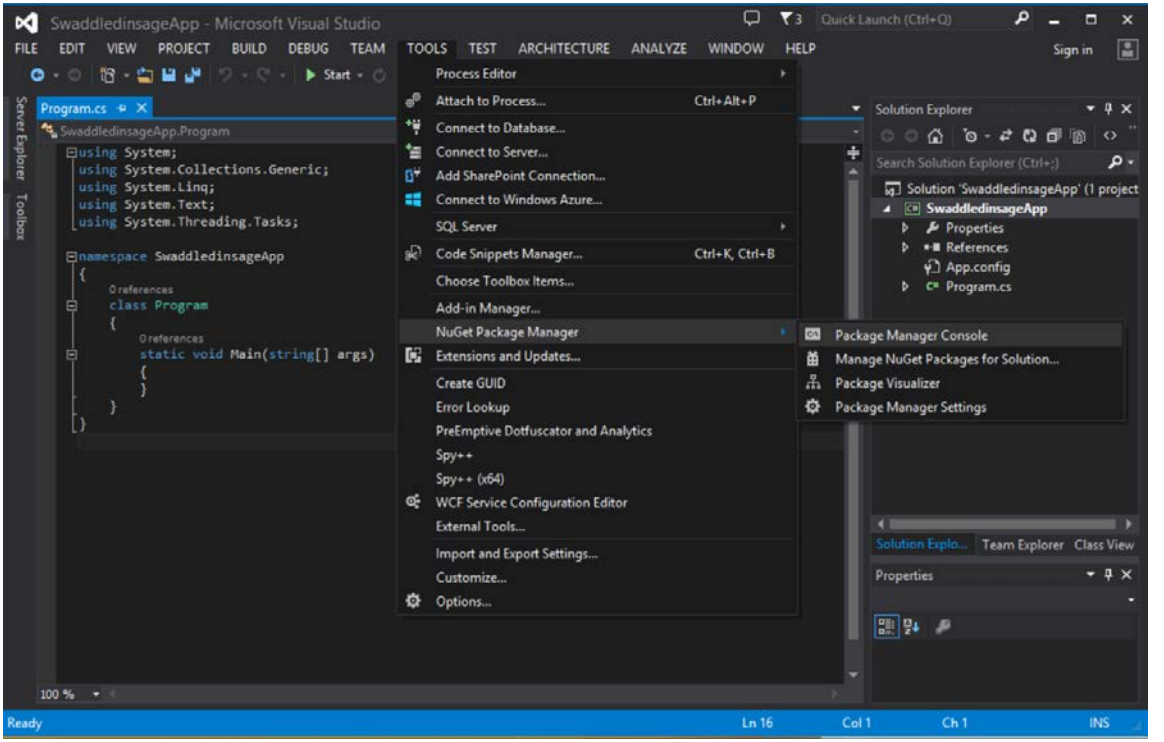


FIGURE 3-4 Launching the Package Manager Console.

After launching the Package Manager Console, Oliver runs the following command to install the HDInsight .NET SDK:

```
Install-Package Microsoft.WindowsAzure.Management.HDInsight
```

Figure 3-5 shows the output from running this command. When installing the package, the command also resolves the dependency for each of the following DLLs, which it adds to the project:

- Microsoft.Hadoop.Client
- Microsoft.WindowsAzure.Management.HDInsight
- Microsoft.WindowsAzure.Management.HDInsight.Framework
- Microsoft.WindowsAzure.Management.HDInsight.Framework.Core

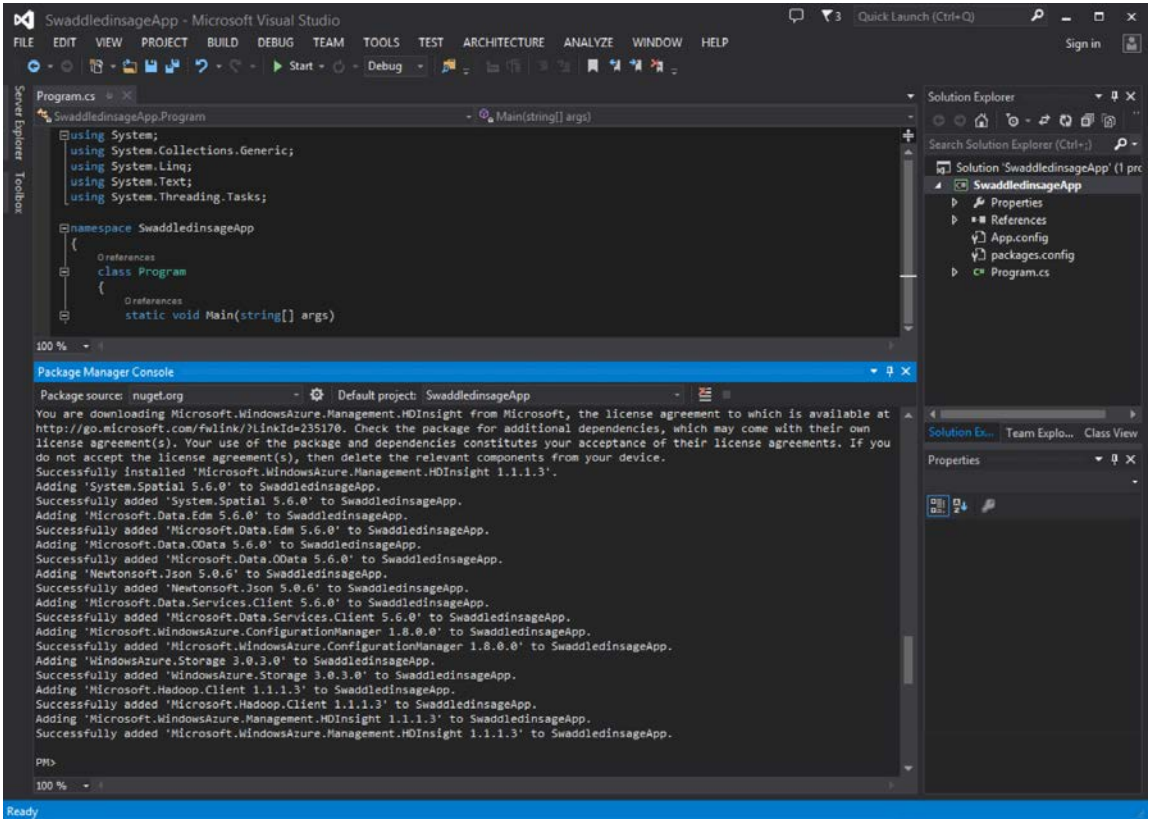


FIGURE 3-5 Installing the HDInsight .NET SDK.

At this point, Oliver has the HDInsight .NET SDK successfully installed on his development machine. He can now use it to submit the MapReduce job to the swaddledinsage HDInsight cluster and the Azure Storage APIs to retrieve the output of the MapReduce job from

storage. The code he uses, shown in Listing 3-8, consists of the following steps:

1. Specify the information for the Azure subscription, HDInsight cluster, and storage account.
2. Get the Azure certificate that has been installed in the certificate store.
3. Create and submit the MapReduce job to the HDInsight cluster.
4. Download the output of the job and write it to standard output.

Tip Using certmgr.msc, you can find the friendly name of the certificate in the Personal certificate store.

LISTING 3-8 Submitting MapReduce job to the HDInsight cluster by using the HDInsight .NET SDK.

```
using System;
using System.Threading.Tasks;
using System.IO;
using System.Security.Cryptography.X509Certificates;

using Microsoft.Hadoop.Client;
using Microsoft.WindowsAzure.Management.HDInsight;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;


    class Program
    {
        static void Main(string[] args)
        {
            // Azure subscription and HDInsight cluster
            string subscriptionID = "<Replace with Windows Azure subscription ID>";
            string clusterName = "swaddledinsage";
            string certfriendlyname = "<Replace with the friendly name for the
certificate>";

            // HDInsight blob storage account
            string storageAccountName = "sagestorage";
            string storageAccountKey = "<Replace with the storage key for the storage
account>";
            string containerName = "swaddledinsage";


            // HDInsight job output
            string downloadFile = "example/data/WordCountOutput/part-r-00000";

            // Get the Azure certificate
```

```
            Console.WriteLine(DateTime.Now.ToLongTimeString() +
                "\tGetting the Windows Azure certificate from the Certificate store");

            X509Certificate2 azureCert = GetAzureCertificate(certfriendlyname);

            // Submit the HDInsight job and wait for the job completion
            Console.WriteLine(DateTime.Now.ToLongTimeString() +
                "\tSubmitting the HDInsight job and start execution");

            Task t = SubmitHadoopJob(subscriptionID, clusterName, azureCert);
            t.Wait();

            // Print the MapReduce job output
            Console.WriteLine(DateTime.Now.ToLongTimeString() +
               "\tGetting the output of the HDInsight job");

            GetHDInsightJobResult(storageAccountName, storageAccountKey,
                        containerName, downloadFile);

            Console.WriteLine("Job completed");
            Console.ReadKey();
        }
}
```

Now Oliver writes various supporting methods that help him submit a MapReduce job to his swaddledinsage HDInsight cluster. The `GetAzureCertificate` method (see Listing 3-9) is used to find and obtain the certificate from the certificate store. The `SubmitHadoopJob` method is used to submit the job to the cluster, and the `GetHDInsightJobResult` method is used to get the results of the MapReduce job.

Note While importing the Azure publish settings file, a certificate is created on the machine where you ran the Azure PowerShell cmdlets. All the certificates created by the Azure PowerShell cmdlets have the value "Windows Azure Tools" in the Issuer field.

LISTING 3-9 The GetAzureCertificate method.

```
/// <summary>
/// Get the Windows Azure certificate
/// </summary>
/// <param name="certfriendlyname">Friendly name for the certificate</param>
/// <returns></returns>
private static X509Certificate2 GetAzureCertificate(string certfriendlyname)
{
    // Get certificate object from certificate store using the friendly name to
identify it
    X509Certificate2 azureCert = null;
```

```
    X509Store store = new X509Store();
    store.Open(OpenFlags.ReadOnly);

    X509Certificate2Collection certCollection =
        store.Certificates.Find(X509FindType.FindByIssuerName,
            "Windows Azure Tools", false);

    // Find the certificate required
    foreach (X509Certificate2 cert in certCollection)
    {
    if (cert.FriendlyName.Equals(certfriendlyname))
    {
        azureCert = cert;
        break;
        }
    }
    return azureCert;
}
```

The name of the certificate is provided in the `Main` method and is defined by the variable `certfriendlyname` . After Oliver successfully finds the certificate, he uses it to create the credentials necessary for creating and submitting jobs to the HDInsight cluster. In the code in Listing 3-10, Oliver invokes the asynchronous method `SubmitHadoopJob`. After the job completes, the output is written to a file in Azure Storage (example/data/WordCountOutput/part-r-00000).

LISTING 3-10 The SubmitHadoopJob method.

```
/// <summary>
/// Create the HDInsight job, and submit it to the HDInsight Cluster
/// </summary>
/// <param name="subscriptionID">Windows Azure subscription ID</param>
/// <param name="clusterName">HDInsight cluster name</param>
/// <param name="azureCert">Certificate issued by Windows Azure</param>
/// <returns></returns>
private static async System.Threading.Tasks.Task SubmitHadoopJob(
    string subscriptionID, string clusterName, X509Certificate2 azureCert)
{

    JobSubmissionCertificateCredential creds =
        new JobSubmissionCertificateCredential(new Guid(
            subscriptionID), azureCert, clusterName);

    // Define the MapReduce job
    MapReduceJobCreateParameters mrJobDefinition = new MapReduceJobCreateParameters()
    {
        JarFile = "wasb:///example/jars/hadoop-examples.jar",
        ClassName = "wordcount"
    };
```

```
    mrJobDefinition.Arguments.Add("wasb:///example/data/sentiment.txt");
    mrJobDefinition.Arguments.Add("wasb:///example/data/WordCountOutput");

    // Create a Hadoop client to connect to HDInsight
    var jobClient = JobSubmissionClientFactory.Connect(creds);


    // Run the MapReduce job
    JobCreationResults mrJobResults =
        await jobClient.CreateMapReduceJobAsync(mrJobDefinition);
}
```

Oliver uses the method `GetHDInsightJobResult`, shown in Listing 3-11, to download the result file using Azure Storage APIs. The content of the file is written to standard output.

LISTING 3-11 The GetHDInsightJob Result method

```
/// <summary>
/// Get the output of the HDInsight job
/// </summary>
/// <param name="storageAccountName">Name of the storage account</param>
/// <param name="storageAccountKey">Storage account key</param>
/// <param name="containerName">Name of the container</param>
private static void GetHDInsightJobResult(string storageAccountName, string
        storageAccountKey, string containerName, string downloadFile)
{
    Stream stream = new MemoryStream();

    CloudStorageAccount storageAccount =
CloudStorageAccount.Parse("DefaultEndpointsProtocol=https;AccountName=" +
storageAccountName + ";AccountKey=" + storageAccountKey);
    CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
    CloudBlobContainer blobContainer =
blobClient.GetContainerReference(containerName);
    CloudBlockBlob blockBlob = blobContainer.GetBlockBlobReference(downloadFile);

    blockBlob.DownloadToStream(stream);
    stream.Position = 0;

    StreamReader reader = new StreamReader(stream);
    Console.WriteLine(reader.ReadToEnd());
}
```

Many other NuGet packages are available for HDInsight. To explore further, Oliver launches **Manage NuGet Packages for Solution** (Tools, Library Package Manager , Manage NuGet Packages for Solution), after typing **HDInsight** in the search box. A list of NuGet packages is displayed based on the search filter criteria. Figure 3-6 shows the various HDInsight-related NuGet packages.

These NuGet packages provide various functionalities, including a .NET MapReduce API and a .NET Language-Integrated Query (LINQ) implementation for working with Hive.

Tip To learn more about LINQ, see http://msdn.microsoft.com/en-us/library/bb397926.aspx.

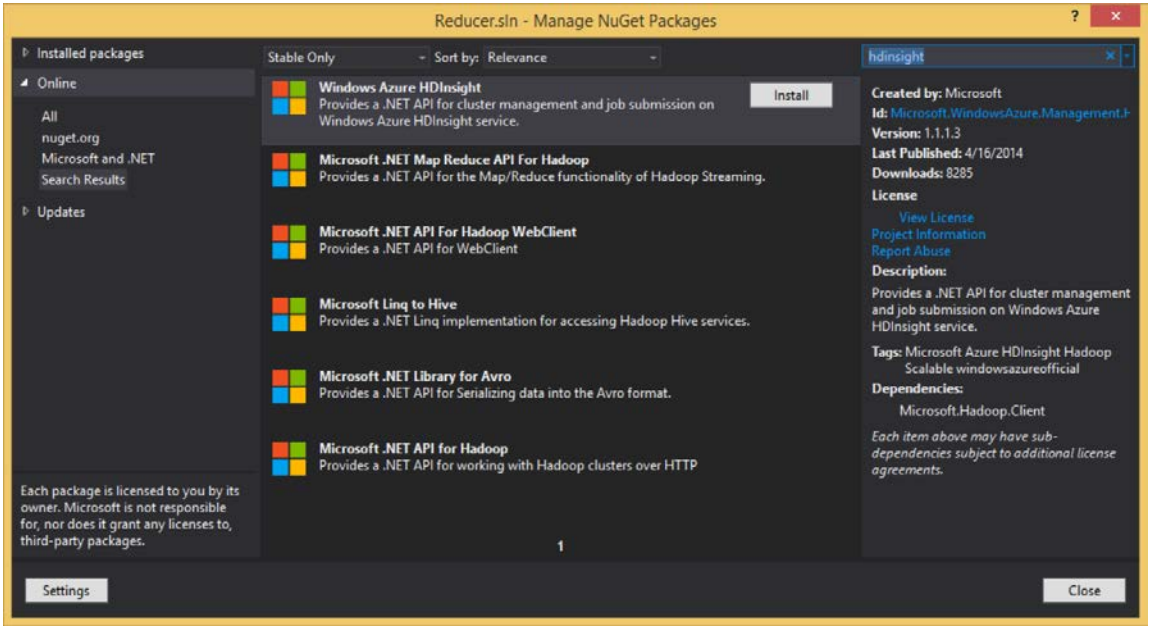


FIGURE 3-6 HDInsight NuGet packages.

Summary

In this chapter, we explored different ways to work programmatically with HDInsight. You learned how the HDInsight PowerShell cmdlets and HDInsight .NET SDK can be used with an HDInsight cluster. In the next chapter, you will learn how to use HDInsight to work with the data in Azure Storage.

Chapter 4

Working with HDInsight data

Natalie is pleased with the convenience and flexible options for working programmatically with a cluster. Now she'd like to know more about the storage side of things. She asks Steve, senior director of data warehousing at *Sage*, to take over the investigation and report back.

Steve's data warehouse team uses many Microsoft business intelligence (BI) tools, and he has wondered how they can use these tools, and Hadoop tools such as Sqoop and Pig, to further improve his team's effectiveness in working with big data. *Sage* itself has also made significant investments in Microsoft BI tools, SQL Server, SharePoint, and database technologies. The company's business analysts such as Cindy use Microsoft Excel and SQL Server Reporting Services for data analysis and reporting. No one at *Sage* is interested in reinventing the wheel. They want to know whether HDInsight can complement *Sage's* existing investments.

In this chapter we introduce you to a few of these Hadoop and Microsoft tools. We'll show you how to use Apache Hive and Apache Pig to analyze data and how to use Apache Sqoop to export data from Azure Storage to an Azure SQL Database. We also describe how to connect to HDInsight by using Excel and the Microsoft Power Query Excel add-in to further refine data and visualize it. After data is loaded in Excel, you can upload the Excel workbook to SharePoint and collaborate and share insights with coworkers in your company.

Using Apache Hive with HDInsight

Steve is thrilled to learn that Apache Hive and the Hive query language (HiveQL) make it possible to quickly and easily process the data that is stored in HDFS or Azure Storage without needing to spend time learning and writing MapReduce programs. Hive handles the MapReduce abstractions itself.

As a database professional his entire career, Steve is very familiar with the Structured Query Language (SQL) and has heard that HiveQL is similar. With HiveQL, he won't have to worry about writing a MapReduce program to analyze data. Under the hood, when the HiveQL statements are executed, Hive runs MapReduce jobs that correspond to the

operations that Steve has expressed in HiveQL. Thus, HiveQL provides Steve with an intuitive and easy way to get started using Hive with HDInsight.

In this first exercise, Steve works with the manager of *Sage*'s online store, Peter, to obtain the web logs from the store's operations. Steve plans to upload the data to an Azure Storage account and use Hive to analyze the web logs, which are in W3C web log format. Steve wants to know who (based on IP addresses) is visiting the company's site and how often, which pages they're visiting, and other online behavior.

Web logs are treasure troves of data about customer behavior. Through proper analysis, organizations can see which pages or other web resources are most popular, how long visitors spend viewing each resource, and much more. This information can inform an organization as to which products are popular, which pages need to be optimized if drop-off rates are high, and so on.

Insights such as these can easily be gleaned from the data by using HiveQL queries. The first step is to tell Hive where to find the data. You can then write a HiveQL query to count the number of web requests made by a specific IP address, for example. In this exercise, Visual Studio is used to upload the log files, and then we use HDInsight cmdlets in PowerShell to create the Hive table.

Upload the data to Azure Storage

Before analyzing the data with Hive, Steve uses Visual Studio to upload the data from the web logs into the Azure blob storage associated with the HDInsight cluster. The web log file Steve uploads is named u_ex1311.log.

Note You can download the sample web log file from http://1drv.ms/OpUzcd.

Figure 4-1 shows the location of u_ex1311.log in Azure. To upload the file, follow these steps:

1. In Visual Studio, in the Server Explorer, choose **Windows Azure.**
2. After expanding the Windows Azure node, choose **Storage.**
3. Right-click **Storage**, and choose **Attach External Storage**. Provide the name of the HDInsight storage account and access key.

> **Note** To understand how to obtain the access key for the storage account, refer to http://www.windowsazure.com/en-us/documentation/articles/storage-manage-storage-account/.

4. After the Azure Storage account is successfully attached, click **Blobs**.

5. In blob view, you will see four icons (shown in Figure 4-1). To upload the web log file, click the Upload icon, the second icon from the left.

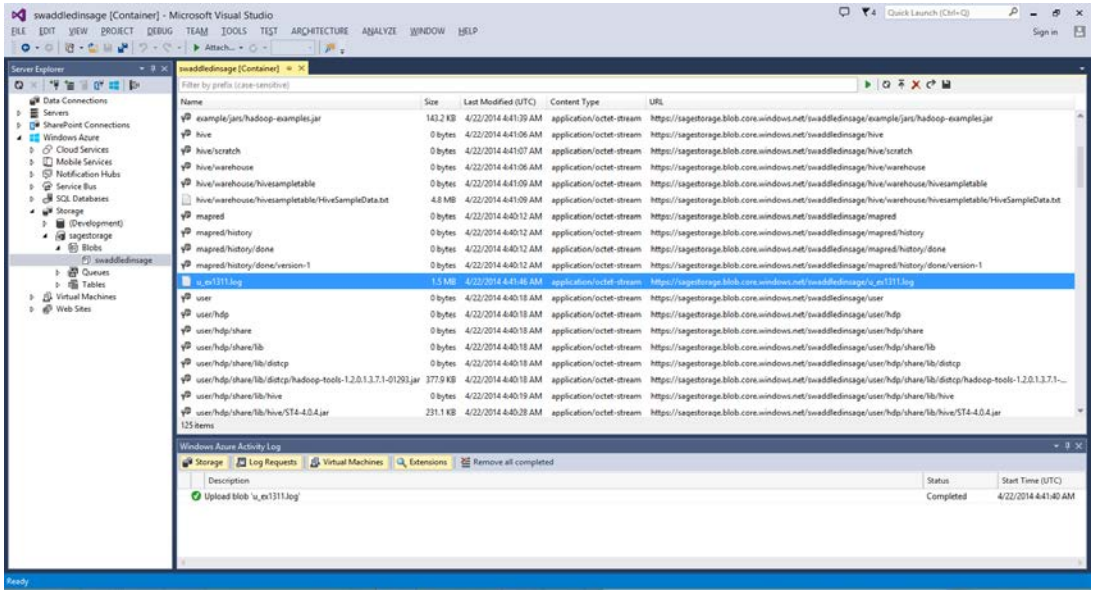


FIGURE 4-1 Uploading the web log file to Azure blob storage using Visual Studio.

Use PowerShell to create tables in Hive

Now that the data from the web logs has been uploaded to Azure Storage, Steve can use HiveQL to create the Hive tables.

HDInsight cmdlets enable you to work with Hive. Steve now uses the PowerShell code shown in Listing 4-1 to create a Hive table and run a query against the data stored in Azure blob storage. The code entails six steps. In the first two steps, Steve specifies the information needed to connect to the Azure subscription, storage account, and HDInsight cluster. In the third step, Steve authors the HiveQL query to create the Hive table.

Tip To install and configure Azure PowerShell for HDInsight, see http://azure.microsoft.com/en-us/documentation/articles/hdinsight-administer-use-powershell/.

In the remaining steps, Steve learns how to use the HDInsight cmdlets to create the HDInsight job definition, start the job, and wait for the job to be completed. To create the HDInsight job definition, Steve specifies the Hive Query (expressed in HiveQL) using the `-Query` parameter.

The Hive table is created in the Hive default location (/hive/warehouse/<*table-name*>). But a Hive table can also be saved in an external location. To create an external Hive table, use the `CREATE EXTERNAL TABLE` command and specify the location of the table by using `STORED AS TEXTFILE LOCATION <location>`, where `location` can be on either HDFS or Azure Storage.

Tip For a discussion of the differences between a Hive-managed table and an external table, see http://blogs.msdn.com/b/cindygross/archive/2013/02/06/hdinsight-hive-internal-and-external-tables-intro.aspx.

Here are the steps involved:

- **Step 1** Specify the name of the Azure subscription and the name of the Azure Storage account.
- **Step 2** Specify the name of the HDInsight cluster.
- **Step 3** Write a HiveQL query to create the table and load the data from Azure blob storage.
- **Step 4** Use the HDInsight cmdlet to create an HDInsight job definition, with the HiveQL query string.
- **Step 5** Start the HDInsight job by using the Hive job definition.
- **Step 6** Wait for the completion of the HDInsight job by using the HDInsight cmdlet `Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $hiveJob -WaitTimeoutInSeconds 3600`.

LISTING 4-1 Create the Hive table

```
# Step 1 - Specify the name of the Azure subscription, and
# the name of the Azure Storage account
$subscriptionName = "Visual Studio Ultimate with MSDN"
$storageAccountName = "sagestorage"
```

```
$containerName = "swaddledinsage"

# Step 2 - Specify the name of the HDInsight cluster
$clusterName = "swaddledinsage"

# Step 3 - HiveQL query that is used to create the table, and load the data from
#    Azure blob storage
$queryString = "DROP TABLE wc3logs; " +
    "CREATE TABLE wc3logs(date string, time string, s_ip string, " +
        " cs_method string, cs_uri_item string,  cs_uri_query string,  " +
        "s_port string, cs_username string, cip string, csUserAgent string, " +
        "csReferer string, scstatus string,  scsubstatus string, " +
        "sc_win32_status string, time_taken string) " +
        "ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '; " +
            "LOAD DATA INPATH " +
    "'wasb://$containerName@$storageAccountName.blob.core.windows.net/u_ex1311.log' "
+
        " OVERWRITE INTO TABLE wc3logs;" +
            "SELECT count(*) FROM wc3logs;"
# Step 4 - Use the HDInsight cmdlet to create a HDInsight job definition, with the
HiveQL
# query string
$hiveJobDefinition = New-AzureHDInsightHiveJobDefinition -Query $queryString

# Step 5 - Start the HDInsight job using the Hive job definition
$hiveJob = Start-AzureHDInsightJob -Subscription $subscriptionName -Cluster
$clusterName -JobDefinition $hiveJobDefinition

# Step 6 - Wait for the completion of the HDInsight job using the HDInsight cmdlet
Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $hiveJob -
WaitTimeoutInSeconds 3600
```

After the PowerShell script runs successfully, the following output is displayed on the screen:

```
StatusDirectory : d9f36906-14ac-4a79-a157-4321bfcee07d+
ExitCode        : 0
Name            : Hive: DROP TABLE wc3logs;
Query           : DROP TABLE wc3logs; CREATE TABLE wc3logs(date string, time string,
s_ip string,
cs_method string, cs_uri_item string,  cs_uri_query string,  s_port string,
cs_username string,
cip string, csUserAgent string, csReferer string, scstatus string,  scsubstatus
string, sc_win32_status string, time_taken string) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ' '; LOAD DATA INPATH
'wasb://swaddledinsage@sagestorage.blob.core.windows.net/u_ex1311.log' OVERWRITE INTO
TABLE wc3logs;SELECT count(*) FROM wc3logs;
State           : Completed
SubmissionTime  : 4/22/2014 4:55:02 AM
Cluster         : swaddledinsage
```

```
PercentComplete : map = 100%,  reduce = 100%
JobId           : job_201404220440_0002
```

As Steve works with HDInsight, he learns how to troubleshoot issues with the PowerShell script. He can use the `Get-AzureHDInsightJobOutput` cmdlet to obtain the output from the HDInsight cluster when the job is executed, as shown here:

```
Get-AzureHDInsightJobOutput -Cluster swaddledinsage -Subscription "Visual Studio
Ultimate with MSDN" -JobId job_201404220440_0002 -StandardError
```

Run HiveQL queries against the Hive table

Now that the Hive table has been created, Steve can use HiveQL to analyze the data. In this exercise, Steve writes a simple HiveQL script to figure out the number of times a specific IP address has accessed the web server. You can go further by geocoding the data (which is not shown here) to determine where the majority of requests are coming from.

The HiveQL script that Steve uses is shown in Listing 4-2.

LISTING 4-2 HiveQL script

```
$subscriptionName = "Visual Studio Ultimate with MSDN"
$storageAccountName = "sagestorage"
$containerName = "swaddledinsage"
$clusterName = "swaddledinsage"

$queryString = "select s_ip, count(*) from wc3logs group by s_ip;"
# This query lists the existing Hive tables

Select-AzureSubscription -SubscriptionName $subscriptionName
Use-AzureHDInsightCluster $clusterName -Subscription (Get-AzureSubscription -
Current).SubscriptionId

Invoke-Hive -Query $queryString
```

In this example, the command `Invoke-Hive` is used to execute the HiveQL statement. When using `Invoke-Hive`, you can also use the `-File` switch to run a HiveQL file that is stored in Azure Storage, as follows:

```
Invoke-Hive –File "wasb://<ContainerName>@<StorageAccountName>/<Path>/myquery.hql"
```

After the query runs successfully, the following output is displayed:

```
Successfully connected to cluster swaddledinsage
Submitting Hive query..
Started Hive query with jobDetails Id : job_201311210542_0021
Hive query completed Successfully
```

```
NULL              4
100.69.12.129   19594
100.69.58.42    6
100.69.62.123  1
22:28:17        1
22:30:20        1
22:57:04        1
23:36:09        1
::1              44
Internet        4
time             4
```

Steve notices that the output contains lines that do not correspond to IP addresses (for example, `::1`). He thinks that the root cause could be the formatting in the log file. When you work with big data, it is common to encounter "dirty" data. There are many options for cleaning the data. Here, Steve uses HiveQL to filter out the erroneous rows by defining a regular expression to identify IP addresses:

```
select length(s_ip), s_ip, count(*) from wc3logs where length(regexp_extract(s_ip,
'(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})',0)) > 0 group by s_ip;
```

After the updated query runs successfully, the following output is displayed:

```
100.69.12.129   19594
100.69.58.42    6
100.69.62.123  1
22:28:17        1
22:30:20        1
22:57:04        1
23:36:09        1
```

Using Apache Pig with HDInsight

As Steve continues his exploration of HDInsight, he learns about Apache Pig and considers how he can use Apache Pig to process data.

Apache Pig provides you with the ability to describe data flows by using a language called *Pig Latin*. Each Pig Latin statement describes how a collection of input rows will be processed and produces a collection of output rows. Pig Latin provides a rich toolbox of arithmetic and relational operators, and you can extend Pig with user-defined functions (UDFs). The design of Pig enables Pig programs to be easily parallelized and provides Pig with the ability to process very large data sets.

In this exercise, Steve uses HDInsight PowerShell cmdlets to execute Pig programs that analyze the W3C web log files and count the number of times a web resource (an image or a webpage, for example) has been accessed. Once again, Steve uploads the web log files to the storage account.

Let's walk through the steps in the Pig script:

- **Step 1** Provides the name of the Azure subscription and the name of the HDInsight cluster (similar to the Hive PowerShell script used earlier).
- **Step 2** Specify the Pig script that will be processed on the HDInsight cluster.
- **Step 3** Create a Pig job definition using the `New-AzureHDInsightPigJobDefinition` cmdlet, and start the Pig job on the HDInsight cluster by using the `Start-AzureHDInsightJob` cmdlet.
- **Step 4** Wait for the completion of the Pig job by using the `Wait-AzureHDInsightJob` cmdlet.
- **Step 5** Display the output of the job

Listing 4-3 shows the code for these five steps.

LISTING 4-3 Pig script

```
# Step 1 - Specify the name of the Azure subscription, and
# the name of the Azure Storage account
$subscriptionName = "Visual Studio Ultimate with MSDN"
$clusterName = "swaddledinsage"

# Step 2 - Specify the Pig script
$pigScript = "LOGS = LOAD 'wasb:///u_ex1311.log' USING PigStorage(' ') as (" +
    "date, time, s_ip,  cs_method, cs_uri_item,  cs_uri_query,  s_port , cs_username,
" +
    "cip, csUserAgent, csReferer, scstatus,  scsubstatus, sc_win32_status,
time_taken); " +
    "BY_ITEM = GROUP LOGS BY cs_uri_item; " +
    "ITEMCOUNT = FOREACH BY_ITEM GENERATE group as cs_uri_item, COUNT(LOGS); " +
    "DUMP ITEMCOUNT; " +
    "STORE ITEMCOUNT into 'wasb:///WebLogPigAnalysis' USING PigStorage(','); "

# Step 3 - Create the HDInsight Pig job definition and start the job
$pigJobDefinition = New-AzureHDInsightPigJobDefinition  -Query $pigScript
$pigJob = Start-AzureHDInsightJob -Subscription $subscriptionName -Cluster
$clusterName -JobDefinition $pigJobDefinition

# Step 4 - Wait for  job completion
Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $pigJob -
```

```
WaitTimeoutInSeconds 3600

# Step 5 - Show the output of the job
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $pigJob.JobId -StandardOutput
```

To see the error output for the HDInsight job, use the `-StandardError` switch with `Get-AzureHDInsightJobOutput`, as shown here:

```
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionName -
JobId $pigJob.JobId -StandardError
```

After the script runs successfully, the following output is displayed:

```
StatusDirectory : 5ff5f626-2d52-4723-9675-e1a236db7f57
ExitCode        : 0
Name            :
Query           : LOGS = LOAD 'wasb:///u_ex1311.log' USING PigStorage(' ') as (date,
time, s_ip,
    cs_method, cs_uri_item, cs_uri_query,  s_port , cs_username, cip, csUserAgent,
csReferer,
    scstatus,  scsubstatus, sc_win32_status,time_taken);
    BY_ITEM = GROUP LOGS BY cs_uri_item;
    ITEMCOUNT = FOREACH BY_ITEM GENERATE group as cs_uri_item,
        COUNT(LOGS); DUMP ITEMCOUNT; STORE ITEMCOUNT into 'wasb:///WebLogPigAnalysis'
USING PigStorage(',');
State           : Completed
SubmissionTime  : 4/22/2014 5:23:11 AM
Cluster         : swaddledinsage
PercentComplete : 100% complete
JobId           : job_201404220440_0015

(/,9801)
(/bp,2)
(/bp/,6)
(Services,4)
(/test.asp,10)
(cs-method,4)
(/iis-8.png,5)
(/test.aspx,9796)
(/bkg-gry.jpg,5)
(/favicon.ico,7)
(/bp/index.php,2)
(/ws8-brand.png,5)
(/msweb-brand.png,5)
(/bp/wp-activate.php,1)
(,8)
```

The built-in function `PigStorage` specifies the delimiter for parsing the data. In this example, a space character is used as the delimiter for parsing the W3C web logs. At the

final step of the Pig script, the `DUMP` operator is used to show the results in standard output. The `STORE` operator can also be used to write the results to storage.

Note To learn more about Apache Pig, see http://pig.apache.org/docs/r0.12.0/start.html.

See Also *You can find a good end-to-end example of analyzing Azure website logs using the PowerShell cmdlets that we have shown in this chapter at http://gallery.technet.microsoft.com/scriptcenter/Analyses-Windows-Azure-web-0b27d413.*

Using Microsoft Excel and Power Query to work with HDInsight data

Microsoft Excel is a familiar application that's used worldwide. Over the years, Excel has evolved from a spreadsheet application to a business-intelligence application. It provides individuals with the ability to combine data from multiple sources, analyze that data, and discover insights. Excel also provides excellent capabilities for visualizing data and sharing insights with coworkers in an organization.

In addition, Microsoft Power BI for Office 365 is a service that provides capabilities and services that enable you to discover, analyze, and visualize data in exciting new ways.

Note To learn about the capabilities in Power BI for Office 365, check out http://office.microsoft.com/en-us/office365-sharepoint-online-enterprise-help/power-bi-getting-started-guide-HA104103589.aspx.

In this exercise, Cindy, *Sage's* business analyst, learns about Power BI and how she can use Power Query for Excel to analyze data. Power Query allows users to find, import, and combine data from multiple public sources, including an HDInsight cluster. It works with many types of files (such as Excel workbooks, CSV, XML, and text), databases, and various nonrelational data sources.

Figure 4-2 shows the various data sources that Power Query can connect to. In this exercise, Cindy uses Power Query to connect to the HDInsight cluster on Azure Storage and the W3C web resource results from the Apache Pig job.

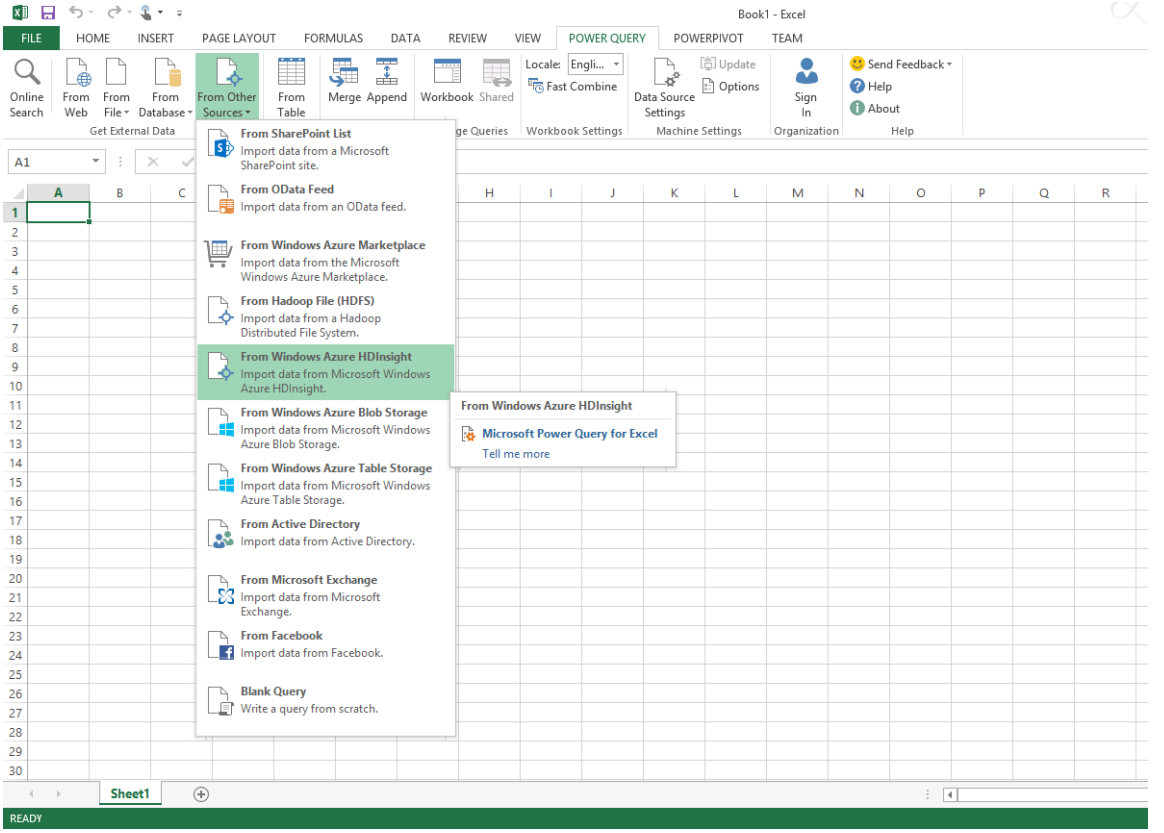


FIGURE 4-2 Power Query and the data sources it can connect to.

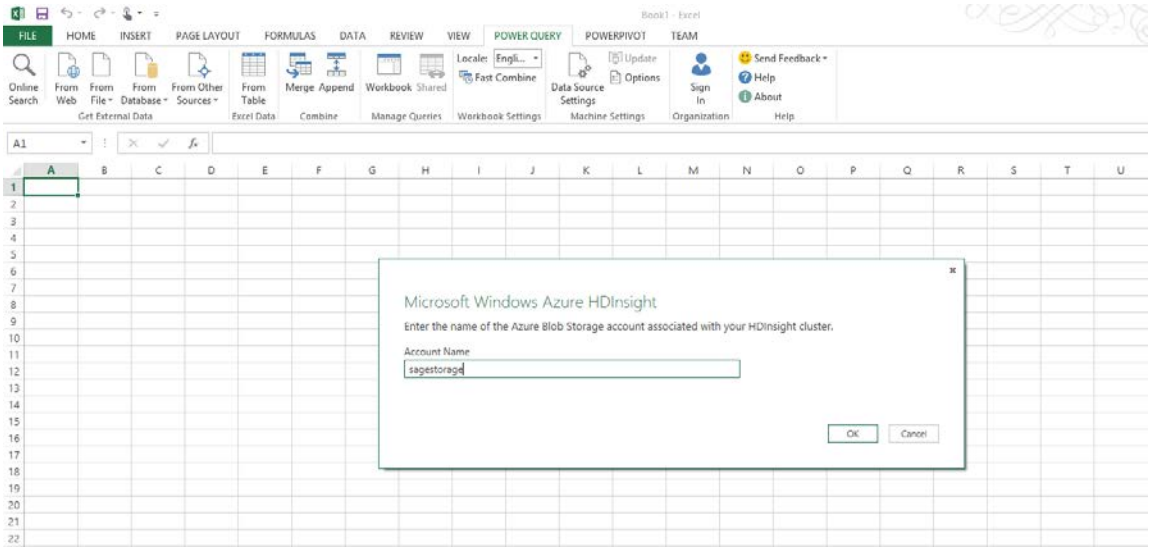
1. To start, Cindy downloads and installs Power Query.

   > **Note** Download the latest version of Microsoft Power Query for Excel from the Microsoft Download Center at http://go.microsoft.com/fwlink/?linkid=286689.

2. After Cindy successfully installs Power Query, she starts Excel and selects the **Power Query** tab.

3. Cindy clicks **From Other Sources**, and then chooses **From Windows Azure HDInsight.**

4. Cindy provides the name for *Sage*'s Azure Storage account that is used by the HDInsight cluster. As shown here, the name of the Azure Storage account is **sagestorage**.

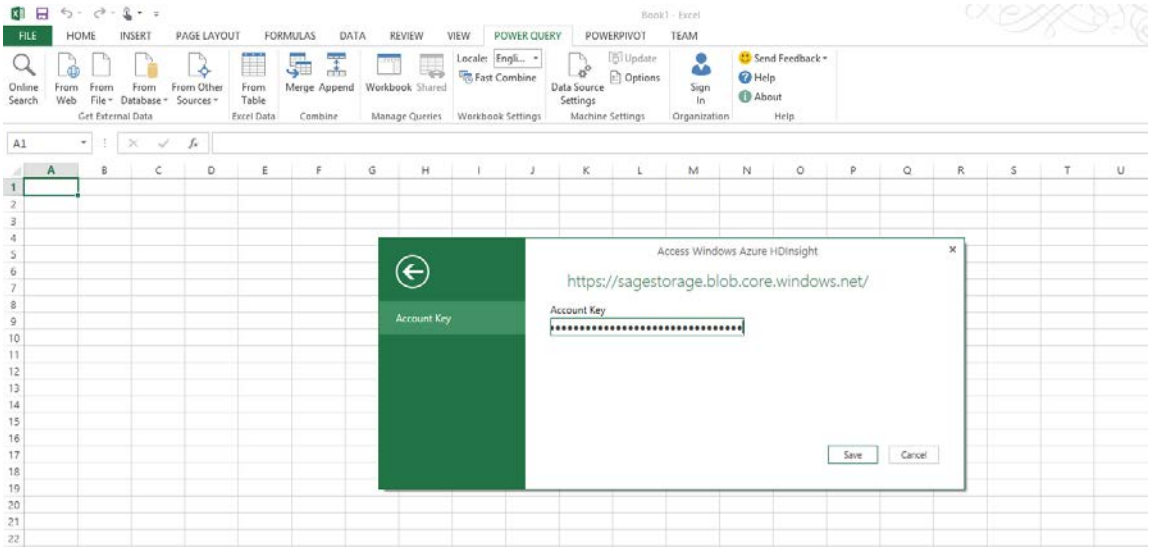


5. Cindy next provides the Azure Storage account key, as shown in the following screenshot. To learn the account key, Cindy works with *Sage's* central IT team. Administrators from the central IT team sign in to the Azure Management Portal to obtain the access key.

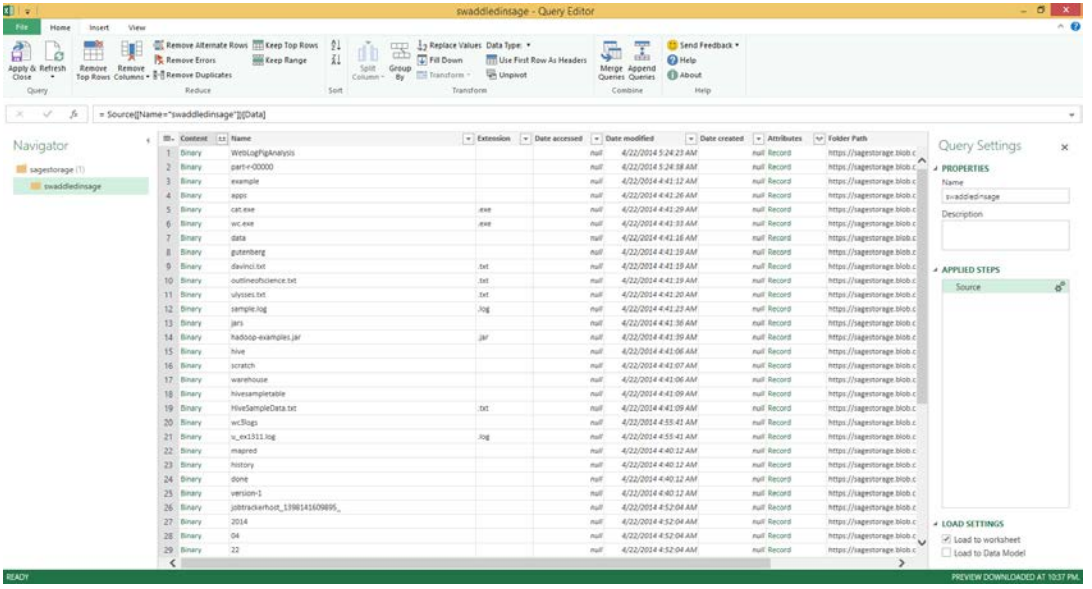
    a. On windowsazure.com, click **Portal** and sign in.

    b. In the management portal, select **All Items** in the left column.

    c. At the bottom of the window, click **Manage Access Keys** to display your keys.



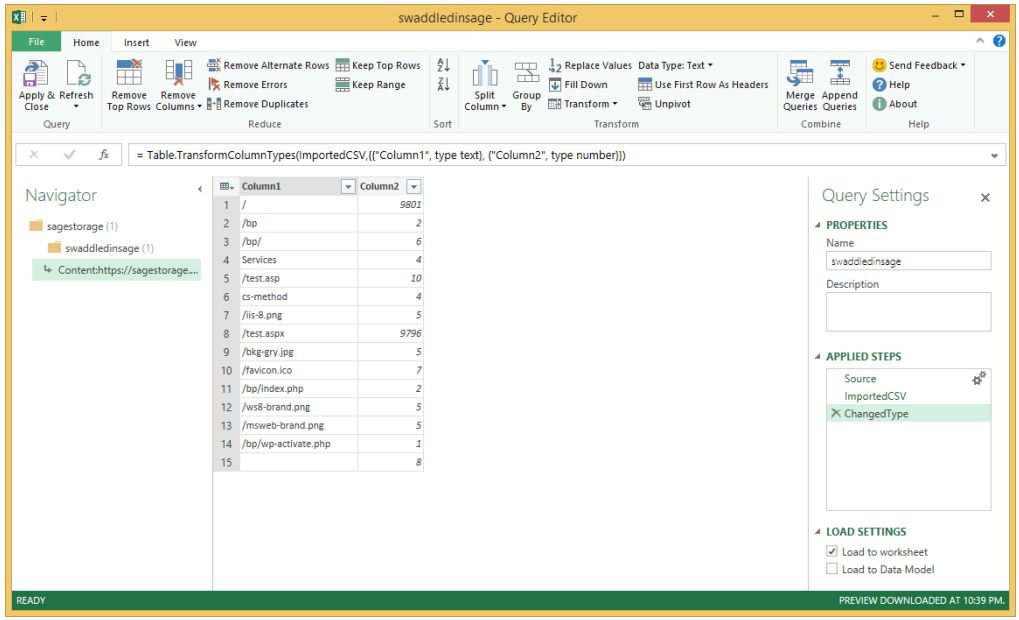
6. After Cindy clicks **Save**, Power Query connects to Azure Storage and displays the data shown in the following screenshot.



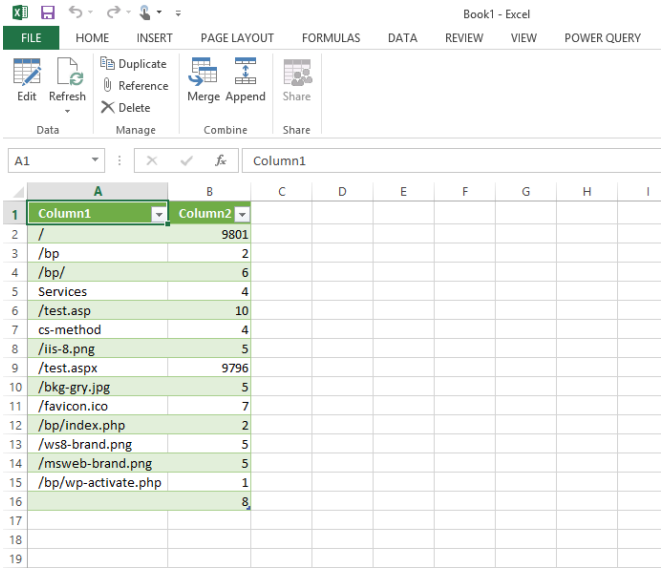
The Navigator pane displays the storage account name and all the containers, tables, and queues in the account. Cindy can highlight a container's name to see its contents. The results from the earlier Pig job are captured in the file part-r-00000, which is stored at the path https://sagestorage.blob.core.windows.net/swaddledinsage/WebLogPigAnalysis/.

> **Note** The file part-r-00000 contains the results from running the Pig script that analyzed the content of the web logs. These results were created by including the line `STORE ITEMCOUNT into 'wasb:///WebLogPigAnalysis' USING PigStorage(',');`.

7. Before selecting the data, Cindy makes sure it is the correct data by peeking at the contents. To see the contents of a file, she clicks **Binary**. The following screenshot shows the contents of part-r-00000. These are the same results obtained from running the Pig job earlier, just displayed in a different format.



8. After Cindy clicks **Apply & Close**, the data from the selected file is added to the Excel worksheet, as shown here.



	A	B
1	Column1	Column2
2	/	9801
3	/bp	2
4	/bp/	6
5	Services	4
6	/test.asp	10
7	cs-method	4
8	/iis-8.png	5
9	/test.aspx	9796
10	/bkg-gry.jpg	5
11	/favicon.ico	7
12	/bp/index.php	2
13	/ws8-brand.png	5
14	/msweb-brand.png	5
15	/bp/wp-activate.php	1
16		8
17		
18		
19		

At this point, Cindy has successfully used Power Query to load data from the HDInsight cluster (specifically, from the storage account associated with the cluster) into an Excel workbook. Now that the data is in Excel, Cindy continues working with the data, enhancing it, combining it with other data, creating reports, sharing, and more.

Because a picture is worth a thousand words, Cindy wants to see trends quickly. She can use a chart to visualize the data that was analyzed in HDInsight. Cindy first renames the columns by right-clicking each of the columns and then clicking **Rename**. She renames the columns URI and NumAccess. Figure 4-3 shows a simple chart that compares the number of occurrences of each resource in the files.

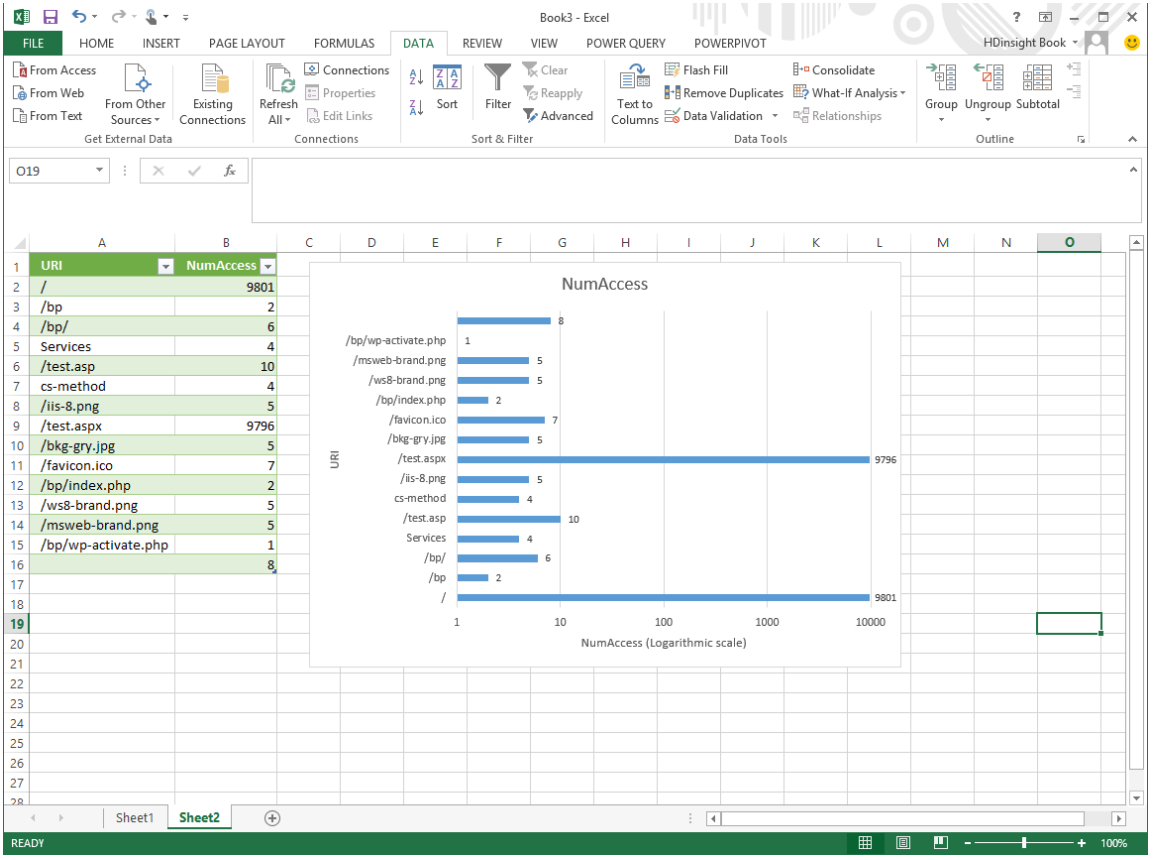


FIGURE 4-3 Visualizing the output of the results from the Pig job.

Using Sqoop with HDInsight

Many organizations have data that is stored in relational database systems (RDBMS). Traditional data warehouses are built to support the needs of business analytics and to provide a repository where insights can be gleaned that offer valuable inputs to business decision makers. BI tools play an important role in making sense of the data, including building OLAP cubes, reports, and data-mining models based on the data in the data warehouse.

As organizations begin implementing a big data strategy, they naturally want to include all their data—whether it is stored in a database management system or stored in Hadoop.

Organizations want and need an end-to-end big data solution that spans their current BI investments and the new world of data.

To bridge the worlds of RDBMS and Hadoop, it is super important to be able to move data between an RDMBS and Hadoop. Sqoop (SQL-to-Hadoop) is an open-source tool that does just that. It allows you to import data from an RDMBS into Hadoop and export data from Hadoop into an RDBMS. When Sqoop moves data between Hadoop and an RDBMS (and vice versa), the data is broken down into smaller pieces and a map-only job is used to move each piece of the data to its destination. Because Sqoop is able to obtain database metadata, it is able to perform type-safe data movement using the data types specified by the metadata.

In this exercise, Steve uses Sqoop and PowerShell to move the results produced by the Pig jobs into an Azure SQL Database. Steve first creates a table (named LogAnalysis) in which to store the results in the Azure SQL Database:

```
CREATE TABLE [dbo].[LogAnalysis](
   [uri] [nvarchar](255) NOT NULL,
   [numaccess] [int] NOT NULL,
 CONSTRAINT [PK_LogAnalysisURI] PRIMARY KEY CLUSTERED ([uri] ASC)
)
```

After the table is created in the Azure SQL Database, Steve creates a synonym for the LogAnalysis table by issuing the SQL command - `CREATE SYNONYM [dbo.LogAnalysis] FOR LogAnalysis.`

Next, Steve runs the Sqoop export from Azure blob storage to the Azure SQL Database. The PowerShell script consists of the following steps:

- **Step 1** Specify the information for the Azure subscription and Azure Storage account.
- **Step 2** Specify the information for the Azure SQL Database. In the example given, the following information is provided:
    - Name of the Azure SQL Database server: doonxon65z.database.windows.net.
    - Name of the database user: swaddledinsageuser@doonxon65z (with the password "mypassword").
    - Name of the Azure SQL Database: swaddledinsage.
    - Name of the table where the data will be exported: LogAnalysis2.

- **Step 3** Create the HDInsight job definition corresponding to the Sqoop export job by using the HDInsight cmdlet `New-AzureHDInsightSqoopJobDefinition`.

  In the HDInsight Sqoop job definition, the keyword `export` is used to specify that the data that is stored in the Azure blob storage account (specified using wasb://) is being exported to the Azure SQL Database.

- **Step 4** Start the HDInsight job.
- **Step 5** Wait for the completion of the HDInsight job.
- **Step 6** Show the output of the HDInsight Sqoop job.

The code is shown in Listing 4-4.

LISTING 4-4 Sqoop export job

```
# Step 1 - Specify the name of the Azure subscription, and
# the name of the Azure Storage account
$subscriptionName = "Visual Studio Ultimate with MSDN"
$clusterName = "swaddledinsage"

# Step 2 - Specify the information for the Azure SQL Database
$sqlDatabaseServerName = "doonxon65z.database.windows.net"
$sqlDatabaseUserName = "steve@doonxon65z"
$sqlDatabasePassword = "mypassword"
$sqlDatabaseDatabaseName = "sagesqldb"
$tableName = "dbo.LogAnalysis"

# Step 3 - Create the Sqoop export job definition
$sqoopDef = New-AzureHDInsightSqoopJobDefinition -Command "export --connect
jdbc:sqlserver://$sqlDatabaseServerName;user=$sqlDatabaseUserName;password=$sqlDatabas
ePassword;database=$sqlDatabaseDatabaseName --columns uri,numaccess --table $tableName
--export-dir wasb:///WebLogPigAnalysis -m 1"

# Step 4 - Start the HDInsight job
$sqoopJob = Start-AzureHDInsightJob -Subscription $subscriptionName -Cluster
$clusterName -JobDefinition $sqoopDef

# Step 5 - Wait for job completion
Wait-AzureHDInsightJob -Subscription $subscriptionName -Job $sqoopJob -
WaitTimeoutInSeconds 3600

# Step 6 - Show the output of the job
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $sqoopJob.JobId -StandardOutput
Get-AzureHDInsightJobOutput -Cluster $clusterName -Subscription $subscriptionname -
JobId $sqoopJob.JobId -StandardError
```

The following output from the HDInsight Sqoop job shows that the Sqoop export job completed successfully and that a total of 15 rows have been written to the LogAnalysis table in the Azure SQL Database:

```
StatusDirectory : c077426c-03bc-46cc-bd09-52b04ecfdfd5
ExitCode        : 0
Name            :
Query           : export --connect
    jdbc:sqlserver://doonxon65z.database.windows.net;user=steve@doonxon65z;
password=mypassword;database=sagesqldb
    --columns uri,numaccess
    --table dbo.LogAnalysis --export-dir wasb:///WebLogPigAnalysis -m 1
State           : Completed
SubmissionTime  : 4/23/2014 5:18:38 AM
Cluster         : swaddledinsage
PercentComplete : map 100% reduce 0%
JobId           : job_201404220440_0081

Warning: HBASE_HOME and HBASE_VERSION not set.
Warning: HBASE_HOME does not exist HBase imports will fail.
Please set HBASE_HOME to the root of your HBase installation.
Warning: HCAT_HOME not set
Warning: HCATALOG_HOME does not exist HCatalog imports will fail.
Please set HCATALOG_HOME to the root of your HCatalog installation.

14/04/23 05:18:50 INFO manager.SqlManager: Using default fetchSize of 1000
14/04/23 05:18:50 INFO tool.CodeGenTool: Beginning code generation
14/04/23 05:18:52 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM
[dbo.LogAnalysis3] AS t WHERE 1=0
14/04/23 05:18:53 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is
c:\apps\temp\hdfs\mapred\local\taskTracker\admin\jobcache\jo
b_201404220440_0081\attempt_201404220440_0081_m_000000_0\work\"C:\apps\dist\hadoop-
1.2.0.1.3.7.1-01293"
14/04/23 05:18:53 WARN orm.CompilationManager: HADOOP_MAPRED_HOME appears empty or
missing
Note: \tmp\sqoop-hdp\compile\b593c16345121b5aa440a5ab077058ad\dbo_LogAnalysis.java
uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/04/23 05:18:54 INFO orm.CompilationManager: Writing jar file: \tmp\sqoop-
hdp\compile\b593c16345121b5aa440a5ab077058ad\dbo.LogA
nalysis3.jar
14/04/23 05:18:54 INFO mapreduce.ExportJobBase: Beginning export of dbo.LogAnalysis3
14/04/23 05:18:57 INFO input.FileInputFormat: Total input paths to process : 1
14/04/23 05:18:57 INFO input.FileInputFormat: Total input paths to process : 1
14/04/23 05:18:57 WARN snappy.LoadSnappy: Snappy native library is available
14/04/23 05:18:57 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/04/23 05:18:57 INFO snappy.LoadSnappy: Snappy native library loaded
14/04/23 05:18:57 INFO mapred.JobClient: Running job: job_201404220440_0082
14/04/23 05:18:58 INFO mapred.JobClient:  map 0% reduce 0%
```

```
14/04/23 05:19:17 INFO mapred.JobClient:  map 100% reduce 0%
14/04/23 05:19:19 INFO mapred.JobClient: Job complete: job_201404220440_0082
14/04/23 05:19:19 INFO mapred.JobClient: Counters: 19
14/04/23 05:19:19 INFO mapred.JobClient:   Job Counters
14/04/23 05:19:19 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=8380
14/04/23 05:19:19 INFO mapred.JobClient:     Total time spent by all reduces waiting
after reserving slots (ms)=0
14/04/23 05:19:19 INFO mapred.JobClient:     Total time spent by all maps waiting
after reserving slots (ms)=0
14/04/23 05:19:19 INFO mapred.JobClient:     Rack-local map tasks=1
14/04/23 05:19:19 INFO mapred.JobClient:     Launched map tasks=1
14/04/23 05:19:19 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=0
14/04/23 05:19:19 INFO mapred.JobClient:   File Output Format Counters
14/04/23 05:19:19 INFO mapred.JobClient:     Bytes Written=0
14/04/23 05:19:19 INFO mapred.JobClient:   FileSystemCounters
14/04/23 05:19:19 INFO mapred.JobClient:     WASB_BYTES_READ=383
14/04/23 05:19:19 INFO mapred.JobClient:     HDFS_BYTES_READ=167
14/04/23 05:19:19 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=77738
14/04/23 05:19:19 INFO mapred.JobClient:   File Input Format Counters
14/04/23 05:19:19 INFO mapred.JobClient:     Bytes Read=0
14/04/23 05:19:19 INFO mapred.JobClient:   Map-Reduce Framework
14/04/23 05:19:19 INFO mapred.JobClient:     Map input records=15
14/04/23 05:19:19 INFO mapred.JobClient:     Physical memory (bytes)
snapshot=181907456
14/04/23 05:19:19 INFO mapred.JobClient:     Spilled Records=0
14/04/23 05:19:19 INFO mapred.JobClient:     CPU time spent (ms)=3187
14/04/23 05:19:19 INFO mapred.JobClient:     Total committed heap usage
(bytes)=514523136
14/04/23 05:19:19 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=671387648
14/04/23 05:19:19 INFO mapred.JobClient:     Map output records=15
14/04/23 05:19:19 INFO mapred.JobClient:     SPLIT_RAW_BYTES=167
14/04/23 05:19:19 INFO mapreduce.ExportJobBase: Transferred 167 bytes in 23.8348
seconds (7.0066 bytes/sec)
14/04/23 05:19:19 INFO mapreduce.ExportJobBase: Exported 15 records.
```

Note To learn about the various Sqoop export arguments, see http://sqoop.apache.org/docs/1.4.4/SqoopUserGuide.html#_literal_sqoop_export_literal.

You can also make use of Sqoop to import data from an Azure SQL Database to an HDInsight cluster. To do this, use the `import` command when creating the HDInsight Sqoop job.

Note To learn about the various Sqoop import arguments, see http://sqoop.apache.org/docs/1.4.4/SqoopUserGuide.html#_literal_sqoop_import_literal.

Summary

As you start implementing a big data strategy in your organization, you need to start thinking about how you can leverage your other data and business-intelligence (BI) assets and use them seamlessly with your new big data tools.

In this chapter, we explored different ways in which an HDInsight cluster can work with the data stored in an Azure Storage account. We started the journey by using Apache Hive and the Hive query language (HiveQL) to analyze data. Then we used Apache Pig and the Pig Latin language to perform a similar task. From these analyses, we gained additional insights on the number of web requests made by a specific IP address. Throughout, we learned how to use the HDInsight cmdlets to create the Hive and Pig jobs and work with them using Azure PowerShell.

Next, we leveraged Microsoft Excel and a new Power BI add-in called Power Query to connect to an HDInsight cluster and access data that has been analyzed by Apache Pig. Power Query can add further value to your data by allowing you to refine the HDInsight data. More importantly, bringing HDInsight data into Excel creates new and exciting possibilities for using the other Microsoft Power BI tools (PowerPivot, Power View, Power Map, and Power Q&A,) to create compelling visualizations and to explore the data further by using natural-language queries.

Finally, we learned how to use Sqoop and HDInsight to move data between HDInsight and an Azure SQL Database. This allows you to bridge two worlds, where you use HDInsight for big data processing and analysis, load the results into a relational database, and use familiar Microsoft BI tools to work with the data.

Chapter 5

What next?

Now that you've learned the basics of working with HDInsight, where do you go from here? And what is the future of the product? How do you integrate an HDInsight cluster into your organization and make it part of your data life cycle?

In this chapter, we'll answer these questions and provide a syllabus you can use to learn more and get started on the next steps of your journey.

Integrating your HDInsight clusters into your organization

Having explored practical applications of HDInsight, now is a good time to see how HDInsight fits in with the rest of your organization. A few years ago, it was common to see Hadoop touted as the only solution to big data. Today, there is broad consensus that while Hadoop is very important, it alone is not the answer to managing big data.

Hadoop—and HDInsight—are critical technologies in a modern big data solution. A good big data solution requires several products and technologies because no single product in the market delivers an end-to-end solution. In our view, a complete big data solution must have three essential elements, illustrated in Figure 5-1:

- A data management layer that manages data of all types and sizes.
- An enrichment layer for data discovery, refinement, and data quality.
- An analytics layer that supports a full range of analytics, including traditional BI, self-service BI, and predictive and real-time analytics.

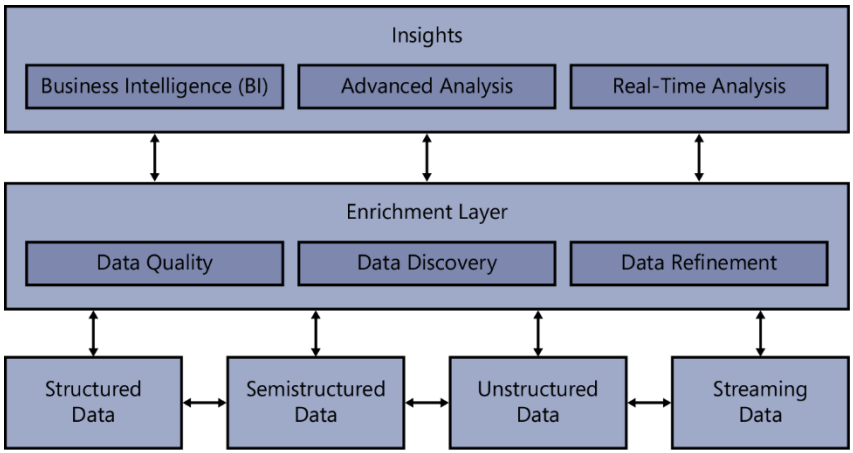


FIGURE 5-1 Essential components of a complete big data solution.

Data management layer

To make the most of all available data, a big data platform must seamlessly manage data of all types, including high-value and low-value data at rest or in motion. It should manage data of all shapes, from highly structured relational data to semistructured data (such as XML) and unstructured data such as PDF files, images, and videos. HDInsight is the best place for storing all your unstructured or semistructured data, and with Microsoft's Sqoop connectors, you can connect your Hadoop cluster on HDInsight to your relational data warehouses. In addition, the new Microsoft Analytics Platform System (formerly known as the Parallel Data Warehouse appliance, or PDW) now combines relational and nonrelational data in the same appliance and ships with both Data Warehouse and HDInsight nodes.

A complete big data solution must manage data of all sizes, from gigabytes to petabytes of data. High-value business data is typically managed in relational data warehouses, while low-value data such as web logs and other exhaust data can be managed cheaply on Hadoop clusters. Streaming data—for example, clickstream data—can be managed with a streaming data engine such as Microsoft StreamInsight. In addition, a great big data platform should break silos by enabling the fluid flow of data across disparate sources within the organization and outside.

Data enrichment layer

Storing and managing data of all types is critical but not sufficient for big data. A good big data solution must also offer the opportunity to enrich data by improving its quality, extend

it with rich data from external sources, and refine it with advanced analytics. Contrary to popular belief, data quality is critical to big data because the old adage is still true—garbage in, garbage out. If you build a model with bad data, the results will be unreliable. Big data does not solve that. In fact, you could argue that big data actually accentuates the need for data quality because it introduces new sources of noncurated data.

So, a good big data solution should provide data-quality tools to convert raw, uncut data into high-quality gems for analysis. Given the abundance of data now available, organizations have a unique opportunity to enrich their analyses by combining their own data with data from third-party providers such as social media, marketing organizations, the US Census Bureau, and the rapidly growing Open Government project.

Microsoft offers several tools for data enrichment. First, SQL Server Integration Services (SSIS) is a great ETL tool for moving and transforming data on-premises. With SSIS, you can move data seamlessly from multiple sources to SQL Server or other database formats. SQL Server Data Quality Services (DQS) is a great tool for cleansing data for analysis. It uses knowledge-based learning to interactively improve the quality of raw data. Master Data Services (MDS), a dedicated master data management (MDM) tool, is another interesting enrichment tool from Microsoft. The Power Query tool in Excel allows you to pull unstructured data easily from several sources, including HDInsight. You can transform the data and merge it with data from other sources—all in Excel.

Analytics layer

The greatest promise of big data is to drive smarter and more actionable decisions based on data of all types and sizes in an organization. To achieve this, a big data solution must deliver actionable insights through a rich set of analytical tools, including business intelligence (BI), advanced analytics (data mining, machine learning, graph mining, and others), and real-time analytics. These insights should be easily accessible to all users—IT, business analysts, operational managers, and strategic decision makers.

Microsoft offers a suite of great products for analytics. First is their corporate BI tools such as SQL Server Analysis Services (SSAS) and SQL Server Reporting Services (SSRS), which can be used for enterprise-wide BI solutions. Through a Hive ODBC driver, you can connect HDInsight to these BI tools and enable your users to analyze unstructured data in the tools they're familiar with. In the past few years Microsoft has also promoted self-service BI, moving BI capabilities such as PowerPivot and Power View into Excel. Second, the new Power Query tool enables users to pull unstructured data from HDInsight into Excel, where they can analyze it easily with all the self-service BI tools. With the new Power Map, users can also map their data in Excel by using its 3-D mapping capability from Bing Maps.

For advanced analytics you have two options. The first is the data mining tools in SQL Server Analysis Services. These tools offer a library of machine learning algorithms for building predictive models (that is, models that predict which customers are most likely to defect or which customers are most likely to buy a given product). They include algorithms such as logistic regression, neural networks, and decision trees. The data mining tools in SQL Server Analysis Services also include clustering algorithms for building customer segmentation models and associative models to predict which products sell well together. These models can be used for up-selling and cross-selling additional products to customers. Professional BI developers can use these algorithms with SQL Server Data Tools. A key benefit of the Microsoft data mining algorithms is programmability: you can make use of the algorithms in your applications by using the Data Mining Extensions (DMX) programming language.

In addition, you can also access the same data mining algorithms from Excel by using the SQL Server Data Mining Add-in for Excel. This add-in offers a simplified user interface and enables you to use, with little technical depth, powerful machine learning algorithms. With this add-in, you can easily build customer segmentation models, do market-basket analysis, and perform many more advanced analytics in Excel.

The second option for advanced analytics is third-party tools such as Mahout. Mahout is an Apache project for machine learning on Hadoop data. Although not supported by Microsoft, Mahout libraries work with HDInsight, so you can deploy and configure Mahout to run on HDInsight. Once Mahout is configured, you can use its libraries to build predictive models using your data on HDInsight clusters. For example, you can use Mahout's recommendation engine to recommend which products will sell well together. Similarly, Mahout’s clustering algorithms can be used to build customer segmentation models using your data in HDInsight. Another option for predictive analytics is the use of solutions from Predixion Software, a Microsoft partner that offers advanced analytics solutions based on the data mining algorithms in SQL Server Analysis Services.

Hadoop deployment options on Windows

Having explored the role of HDInsight in a big data solution, let's examine your options for using Hadoop on Windows. There are now three ways to deploy Hadoop on Windows, as illustrated in Figure 5-2: in the cloud, in an appliance, or on your own servers. HDInsight on Azure, which has been discussed extensively in this book, is great if you want the full benefits of the cloud, such as low cost or elastic scalability. It is also ideal for those with data born in the cloud, such as web clickstreams or data from social media sites.

If you prefer to deploy Hadoop on-premises, you have two options: you can use HDInsight in an appliance or on your own servers. Both options are great if you have large volumes of data generated on-premises. They are also useful if you want full control over your data because you will deploy your Hadoop clusters in your own data centers.

The latest release of Parallel Data Warehouse, now called Microsoft Analytics Platform System, has both data warehouse and Hadoop servers in the same appliance. The Hadoop servers run HDInsight. This enables you to deploy both Hadoop and a traditional data warehouse in the same appliance. The appliance form factor also reduces the administrative burden because it ships preconfigured and pretuned. Hence, you don't have to deploy and provision HDInsight from scratch. We cover the appliance form factor more later in this chapter.

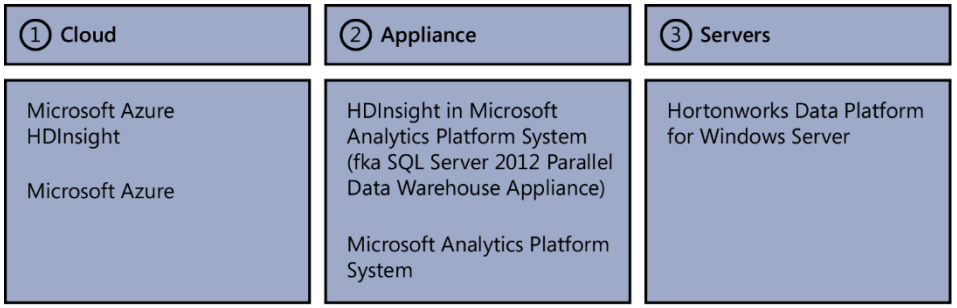


FIGURE 5-2 Three Hadoop deployment options on Windows.

Finally, for those who prefer to deploy Hadoop on their own servers on-premises, we recommend Hortonworks Data Platform (HDP) for Windows. Distributed by Hortonworks, HDP for Windows is a 100 percent Apache Hadoop distribution that runs Hadoop as a first-class citizen on Windows Server. Hortonworks collaborated very closely with Microsoft to port the Hortonworks Data Platform from Linux to Windows Server. As a result, this distribution offers interoperability across Windows and Linux. Because there is symmetry between HDP on Linux and Windows, you can easily port your Hadoop application across Linux and Windows Server. Figure 5-3 shows the Hadoop projects in Hortonworks Data Platform for Windows.

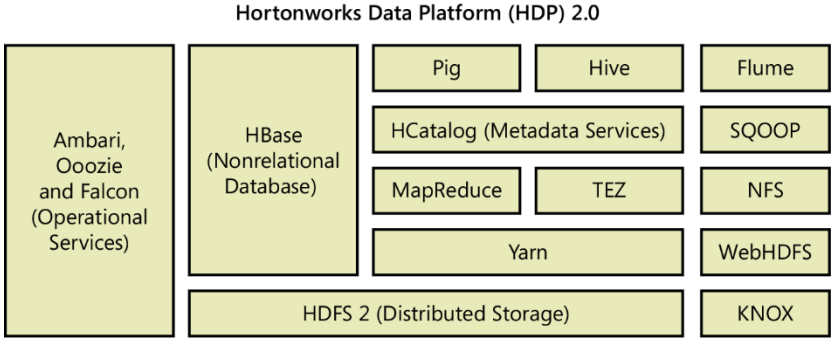


FIGURE 5-3 Components of Hortonworks Data Platform (HDP) 2.0.

Latest product releases and the future of HDInsight

Now let's take a look at the latest product releases and the exciting road ahead for HDInsight. Microsoft has made important improvements to HDInsight both for use in the cloud and for on-premises use. First, we'll examine HDInsight improvements on Azure, and then we'll cover HDInsight in the Parallel Data Warehouse appliance.

Latest HDInsight improvements

As you have seen so far, HDInsight is a powerful Hadoop distribution that opens new opportunities for developing Hadoop applications in the cloud. With HDInsight, you can easily deploy a Hadoop cluster in less than 20 minutes, which is impressive. You can also develop Hadoop applications with Java, .NET, or other languages. That said, managing HDInsight or deploying Hadoop jobs involves a great deal of scripting, which can become complex.

Microsoft engineers are working hard to improve HDInsight on several fronts. In this section, we discuss two of these improvements: a simplified user experience and migrating to Hadoop 2.0.

First, the development of a new graphical user interface is underway. When this interface is released, you can expect to use it to easily submit your Hadoop jobs, so you won't have to depend only on the SDK or PowerShell scripts. This improvement, coupled with the graphical user interface for cluster monitoring, will significantly simplify the user experience on HDInsight.

The second important improvement for HDInsight is support for Hadoop 2.0. Last

October, Hortonworks released Hortonworks Data Platform 2.0 (aka HDP 2.0). Based on Apache Hadoop 2.0, HDP 2.0 is a major Hadoop release that delivers YARN, Project Stinger, and real-time stream processing through STORM. Figure 5-3 shows the key components of HDP 2.0.

YARN is a major step forward for Hadoop because it allows Hadoop to support new workloads beyond MapReduce patterns. With YARN, you can run jobs for graph mining to glean insights from social network sites such as Twitter and Facebook. YARN also acts as an operating system for Hadoop 2.0 because it provides support for several users running different workloads at the same time. The prospect of many concurrent users running multiple workload types on YARN makes Hadoop 2.0 (and therefore HDP 2.0) very powerful and promising for enterprise users.

HDP 2.0 also offers phase 2 of Project Stinger, an open-source initiative led by Hortonworks and several partners, including Microsoft and Facebook. Through this project, the Hadoop community plans to improve the performance and scalability of Hive to enable users to run interactive instead of batch-only queries. Project Stinger also enables support for SQL queries.

Finally, by supporting STORM, HDP 2.0 will run streaming queries. This overcomes one of the biggest limitations of Hadoop version 1.0, which runs only in batch mode. These improvements combined now enable HDP 2.0 to run in batch, interactive, and real-time modes, which is pretty powerful.

Microsoft has been firing on all cylinders and making rapid progress on the adoption of Hadoop 2.0. In February 2014, Microsoft announced a preview of HDInsight. This latest version of HDInsight is based on HDP 2.0, which now supports Apache Hadoop 2.2. By the time this book is published, this latest release of HDInsight should be generally available.

HDInsight and the Microsoft Analytics Platform System

Having explored the future of the HDInsight service on Azure, let's turn our attention to HDInsight on-premises. Microsoft has just announced a new release of its data warehouse appliance, named Microsoft Analytics Platform System, which ships with HDInsight servers. But as background before we delve into this new appliance, let's review PolyBase, a critical and interesting big data technology that was first released in SQL Server 2012 Parallel Data Warehouse in March 2013.

PolyBase seamlessly combines structured and unstructured data and was developed by Microsoft in collaboration with Dr. David Dewitt and his team at the Jim Gray System Labs. PolyBase simplifies big data for database professionals and developers by enabling users to

query both relational and nonrelational data with T-SQL queries. Users don't need to understand MapReduce to query the data. For example, suppose you want to get the sales of a given car in the last three quarters and you also want to know customers' sentiments toward your cars. Also assume that you store Twitter feeds in a Hadoop cluster. To collect this data, you need to configure PolyBase to point to your Hadoop cluster and then provide two pieces of information: the name URL and the login details for your Hadoop cluster.

With this information supplied, you can send your queries to PolyBase via T-SQL statements. When PolyBase receives the query, it fetches the relational data from the Parallel Data Warehouse appliance and the nonrelational data from your Hadoop cluster. If needed, PolyBase also performs joins before sending the answer in the result set. Figure 5-4 is a simple schematic diagram that shows how PolyBase works. PolyBase supports Hortonworks Data Platform 1.3 on Linux, Hortonworks Data Platform 1.3 for Windows Server, and even Cloudera 4.3, which is a competitive Hadoop distribution.

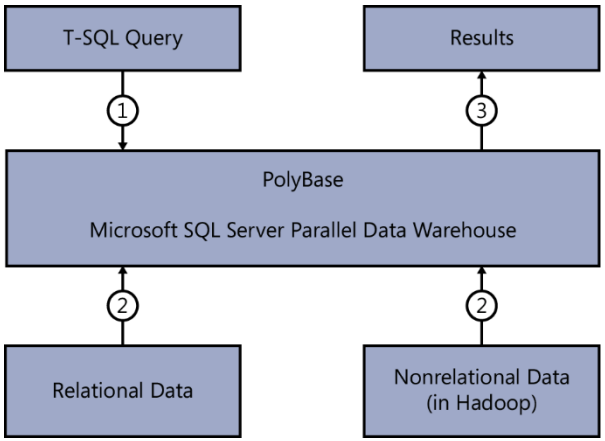


FIGURE 5-4 PolyBase in action.

In the first release of PolyBase, the Hadoop cluster was outside the Parallel Data Warehouse appliance. However, the new release of Parallel Data Warehouse (Microsoft Analytics Platform System) includes HDInsight nodes inside the appliance. In the new release, each appliance has a combination of data warehouse and HDInsight nodes. However, these nodes run on separate servers—in other words, each node in the appliance runs either data warehouse or HDInsight functionality. Each appliance also includes PolyBase, which combines relational and nonrelational data from the data warehouse or HDInsight nodes. In addition, PolyBase also continues to support other Hadoop distributions outside the appliance, such as Hortonworks Data Platform 1.3 on Linux, Hortonworks Data Platform 1.3 for Windows Server, and Cloudera 4.3. Furthermore, in Microsoft Analytics

Platform System, PolyBase supports Azure blob storage, which means it can also pull data directly from Azure Storage and combine it with relational data in the appliance.

Figure 5-5 shows a simple schematic diagram of Microsoft Analytics Platform System. Each PDW appliance comes with dedicated storage servers, a control node that is the nerve center of massively parallel processing of the appliance, a management server for managing the appliance, and a landing zone for loading data into the appliance. In addition, the appliance ships with servers for HDInsight. For simplicity, Figure 5-5 shows PolyBase only with the Parallel Data Warehouse and HDInsight nodes. The exact number of PDW and HDInsight nodes in each appliance depends on the vendor. Note that PDW appliances are offered by Dell, HP, and Microsoft. For all vendors, however, each appliance will have an equal number of PDW and HDInsight nodes.

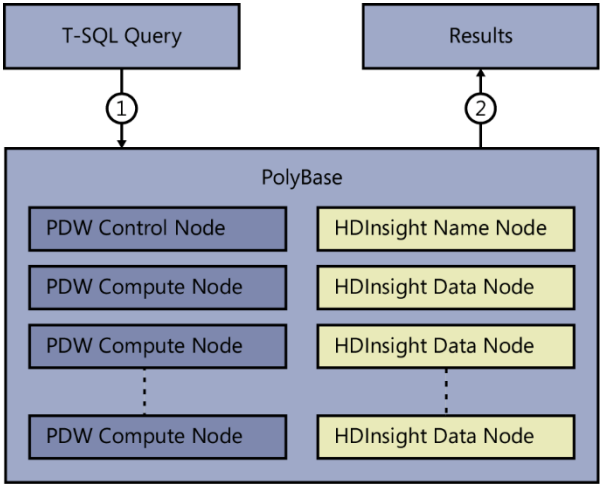


FIGURE 5-5 Schematic diagram of the new Microsoft Analytics Platform System.

The Analytics Platform System supports three key usage scenarios:

- HDInsight as a staging area for PDW
- Incremental loading and reporting on HDInsight
- Hadoop as a store for cold data

Let's discuss these three scenarios in more detail.

Data refinery or data lakes use case

In this scenario, illustrated in Figure 5-6, HDInsight is used as a staging area for PDW. Raw data is stored in the HDInsight region of the appliance, and all users access the data through the PDW region. DBAs can access the data with the usual client BI tools for the Analytics Platform System, such as SQL Server Analysis Services, or by using third-party BI tools. Information workers such as business analysts can access the data through self-service BI tools, including Power Pivot and Power Query in Excel. This usage pattern is commonly referred to as the data refinery or data lakes use case, in which Hadoop is used as a transient area where raw data is refined before storage and analysis in a data warehouse.

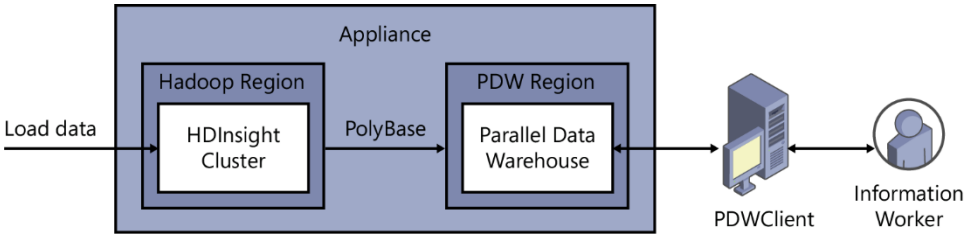


FIGURE 5-6 HDInsight as a staging area for PDW.

Data exploration use case

In this use case, the Hadoop cluster is used as an active store where data is loaded incrementally from multiple sources. Hadoop developers can load data directly onto the HDInsight cluster in the appliance and can also run Hadoop jobs directly on the same cluster. End users can run reports on the data in the HDInsight cluster using Hadoop applications or BI tools from Microsoft or third parties. Figure 5-7 illustrates this usage pattern.

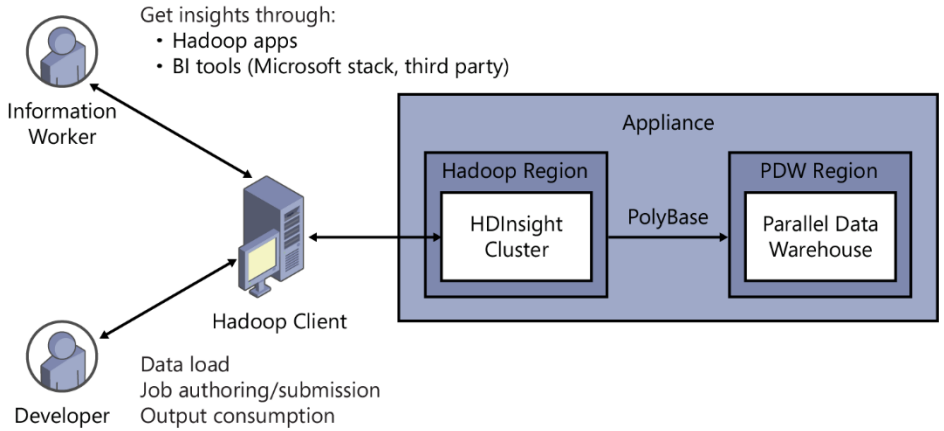


FIGURE 5-7 Incremental loading into Hadoop.

Hadoop as a store for cold data

Unlike the previous two scenarios, which use HDInsight as a store for active data, this usage scenario uses HDInsight as a store for cold data. In this scenario, hot data (data that's frequently used to run a business) is stored in the PDW region. When the data goes cold (when it is no longer needed to run the business on a daily basis), you can move it from the PDW region and store it on your Hadoop cluster in the HDInsight region of the appliance. Of course, you can still access the cold data if needed by using Hadoop applications or BI tools. This usage scenario is illustrated in Figure 5-8.

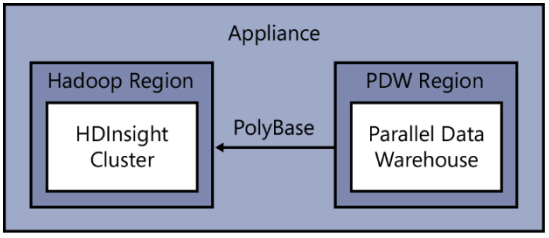


FIGURE 5-8 Hadoop as a cold store.

Study guide: Your next steps

This section provides a study guide to help you deepen and expand your learning of HDInsight and related technologies. The Azure Documentation Center is a great source of

learning material on HDInsight, so let's start with the documentation available on the landing page for HDInsight documentation at http://azure.microsoft.com/en-us/documentation/services/hdinsight/.

Getting started with HDInsight

This tutorial is a good introduction to HDInsight from Microsoft. It shows how to quickly install and provision HDInsight, introduces PowerShell scripts for HDInsight, and shows you how to connect to Microsoft BI tools. For more details visit http://azure.microsoft.com/en-us/documentation/articles/hdinsight-get-started/.

Running HDInsight samples

Before writing your first MapReduce job, it is often useful to see examples of jobs written by others. Luckily, HDInsight currently contains four sample MapReduce jobs—each one demonstrates how to run a specific type of job on a Hadoop distribution. These samples demonstrate the programming flexibility of HDInsight, as some of them are written in Java while another is written in C#. By running these samples yourself, you can see how HDInsight supports not only .NET but also Java applications. The examples are:

- **Pi estimator** Estimates the value of Pi using a MapReduce program
- **Word Count** Counts the frequency of words in a text file
- **10-GB GraySort** Runs the GraySort MapReduce job on a 10-GB file. The Sort Benchmark measures the sorting performance of different platforms. The GraySort in particular is a benchmark whose metric is the sort rate (in TB/min) for sorting large data volumes. The GraySort sample in HDInsight uses a 10-TB dataset with MapReduce applications developed by Arun Murthy and Owen O'Malley that won the Daytona GraySort benchmark in 2009. More details of the Sort Benchmark are available at http://sortbenchmark.org/.
- **C# streaming sample** Shows how to write a MapReduce job in C# and run it using Hadoop streaming.

Chapter 2, "Getting started with HDInsight," shows how to run the Word Count sample. You can learn more about this and the other three samples at http://azure.microsoft.com/en-us/documentation/articles/hdinsight-run-samples/.

Connecting HDInsight to Excel with Power Query

If you want to learn more about moving data from HDInsight into Excel with Power Query, you'll find a great tutorial at http://azure.microsoft.com/en-us/documentation/articles/hdinsight-connect-excel-power-query/. This tutorial shows how to install Power Query, connect it to your Hadoop cluster in HDInsight, and move data from your cluster into Excel.

Using Hive with HDInsight

Chapter 3 provided a good introduction to programming HDInsight with Hive, Pig, PowerShell, and C#. For more information on Hive, see the tutorial at http://azure.microsoft.com/en-us/documentation/articles/hdinsight-use-hive/.

There is also a good tutorial on Pig programming at http://azure.microsoft.com/en-us/documentation/articles/hdinsight-use-pig/.

To learn more about submitting your jobs programmatically on HDInsight, visit http://azure.microsoft.com/en-us/documentation/articles/hdinsight-submit-hadoop-jobs-programmatically/.

Hadoop 2.0 and Hortonworks Data Platform

Through its Hortonworks University program, Hortonworks offers several training courses on Hadoop 2.0 with an emphasis on the Hortonworks Data Platform. Their courses range from an introduction through cluster administration, solution development, and even data science. In addition, Hortonworks has a certification program on the Hortonworks Data Platform. More information is available at http://hortonworks.com/hadoop-training/.

PolyBase in the Parallel Data Warehouse appliance

There are several useful learning materials on the PolyBase technology in the new Microsoft Analytics Platform System (formerly known as the Parallel Data Warehouse appliance, or PDW). Here is a starting point:

- http://gsl.azurewebsites.net/Projects/Polybase.aspx
- http://www.youtube.com/watch?v=ScLWU6NmLd4
- http://www.youtube.com/watch?v=AKS1u_KO7AA

Recommended books

- Tom White's book *Hadoop: The Definitive Guide* is a great primer for Hadoop. You should check to be sure it is updated for Hadoop 2.0. This book is available on Amazon: http://www.amazon.com/Hadoop-Definitive-Guide-Tom-White/dp/1449311520/ref=sr_1_3?s=books&ie=UTF8&qid=1386428765&sr=1-3&keywords=hdinsight.
- We also recommend the book *Programming Hive* by Edward Capriolo, Dean Wampler, and Jason Rutherglen. It is a great book for those who want to learn about Hive in depth. More details at http://www.amazon.com/Programming-Hive-Edward-Capriolo/dp/1449319335/ref=sr_1_1?s=books&ie=UTF8&qid=1386611932&sr=1-1&keywords=Programming+Hive.
- Finally, for those keen to learn about YARN and Hadoop 2.0, we recommend *Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2* by Arun Murthy, Jeffrey Markham, Vinod Vavilapalli, and Doug Eadline. Arun Murthy is one of the leaders of the Apache Hadoop 2.0 project. More details are available at http://www.amazon.com/Apache-Hadoop-YARN-Processing-Addison-Wesley/dp/0321934504.

Summary

In this chapter we explored the role of HDInsight in a complete enterprise big data solution. In particular we examined the three options for deploying Hadoop on Windows, including deployment on-premises and in the cloud. Second, we explored the future of HDInsight on Microsoft Azure and in an appliance—the Microsoft Analytics Platform System. Finally, we provided a study guide to expand your learning of HDInsight and related technologies.

About the authors

Avkash Chauhan is the founder and principal at Big Data Perspective, working to build a product that makes Hadoop accessible to mainstream enterprises by simplifying its adoption, customization, management, and support for a Hadoop cluster. While recently at Platfora, he participated in building big data analytics software that runs natively on Hadoop. Previously he worked eight years at Microsoft building cloud and big data products and providing assistance to enterprise partners worldwide. Avkash has more than 15 years of software development experience in cloud and big data disciplines. He is an accomplished author, blogger, and technical speaker and loves the outdoors.

Valentine Fontama is a principal data scientist in the Data and Decision Sciences Group at Microsoft. Val has more than eight years of data science experience. After obtaining his PhD in neural networks, he was a new technology consultant at Equifax in London, where he pioneered the application of data mining in the consumer credit industry. Over the last seven years, Val was a senior product marketing manager for big data and predictive analytics in SQL Server marketing, responsible for machine learning, HDInsight, Parallel Data Warehouse, and Fast Track Data Warehouse. Val also holds an MBA in strategic management and marketing from the Wharton School, an MS in computing, and a BS in mathematics and electronics. He has published 11 academic papers and is an accomplished speaker about big data.

Michele Hart is a senior technical writer with more than 20 years writing experience, the last 6 at Microsoft. She has written countless knowledgeable words for various industries, including finance, entertainment, Internet, telecom, and education. She spent several years as a manager and director of writing, training, and support teams, several more years as a stay-at-home mom, and the last eight or so as an individual contributor focusing on SQL Server and Power BI articles and videos.

Wee-Hyong Tok is a senior program manager on the SQL Server team at Microsoft. Wee-Hyong has a range of experiences working with data, with more than six years of data platform experience in industry and six years of academic experience. After obtaining his PhD in data streaming systems from the National University of Singapore, he joined Microsoft and worked on SQL Server Integration Services (SSIS). He was responsible for shaping the SSIS Server, bringing it from concept to its inclusion in SQL Server 2012. Wee-Hyong has published 20 academic papers and speaks regularly at technology conferences.

Buck Woody is a senior technical specialist for Microsoft, working with enterprise-level clients to develop computing platform architecture solutions within their organizations. With more than 25 years of professional and practical experience in computer technology, he is also a popular speaker at TechEd, PASS, and many other conferences. Buck is the author of more than 500 articles and five books on databases and teaches a database design course at the University of Washington.

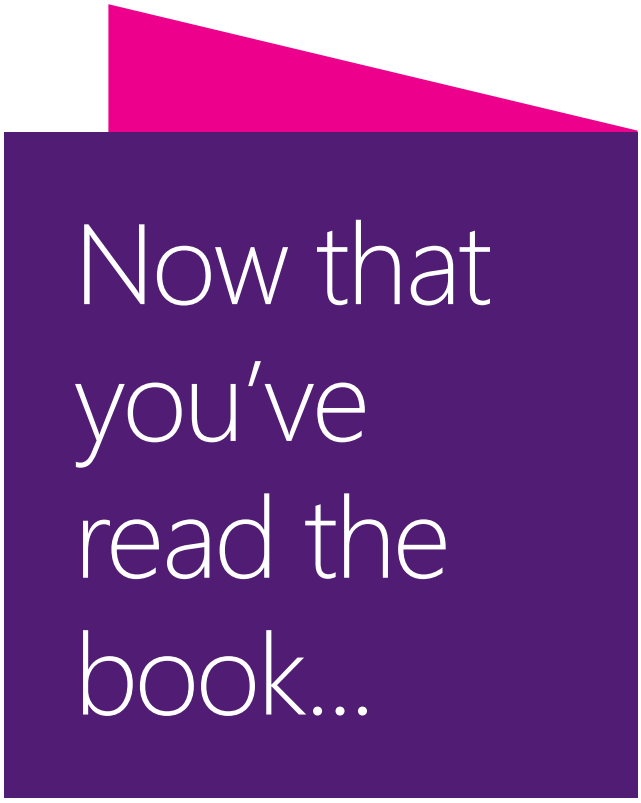
From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

www.microsoftvirtualacademy.com/ebooks

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

Microsoft Press



Tell us what you think!

Was it useful?
Did it teach you what you wanted to learn?
Was there room for improvement?

Let us know at http://aka.ms/tellpress

Your feedback goes directly to the staff at Microsoft Press,
and we read every one of your responses. Thanks in advance!

Microsoft