

AI and Big Data on IBM Power Systems Servers

Ivaylo B. Bozhinov

Anto Ajay Raj John

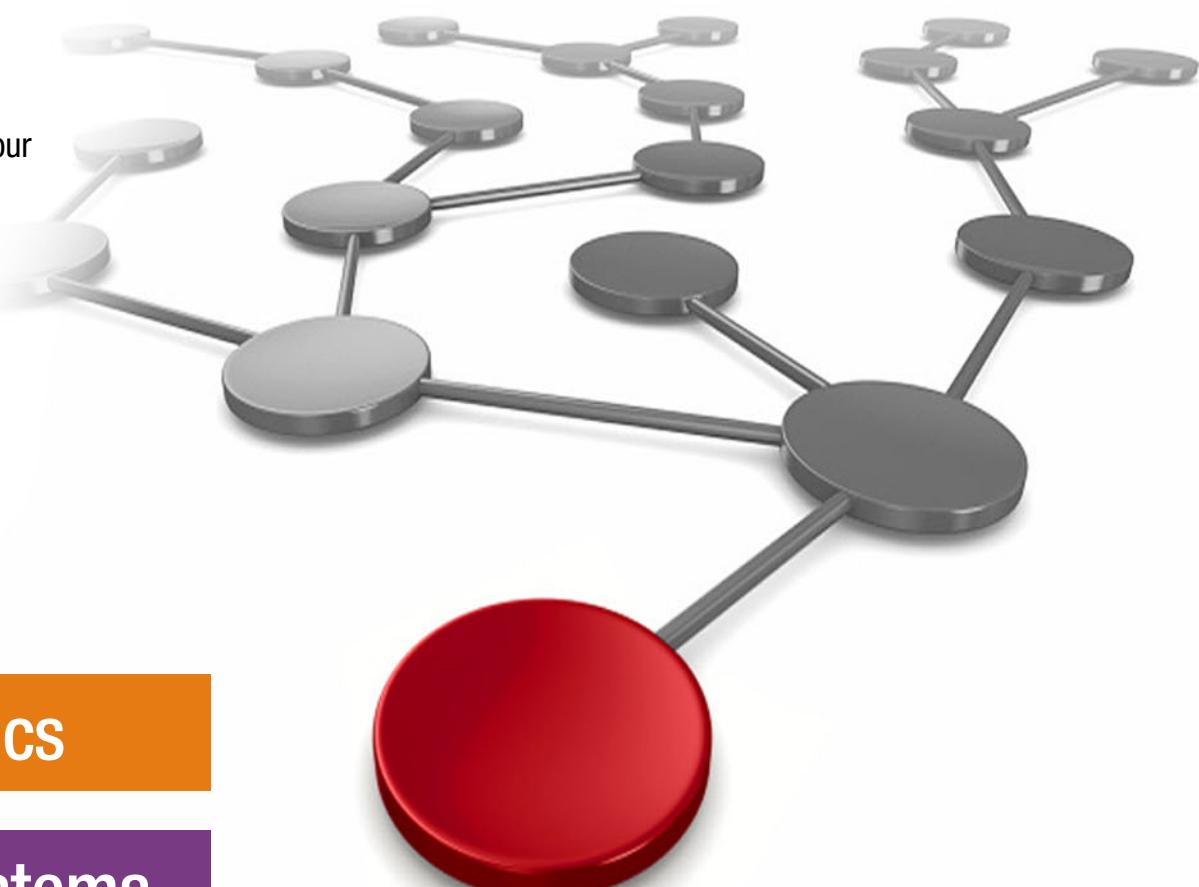
Rafael Freitas de Lima

Ahmed (Mash) Mashhour

James Van Oosten

Fernando Vermelho

Allison White



 Analytics

Power Systems



IBM Redbooks

AI and Big Data on IBM Power Systems Servers

March 2019

Note: Before using this information and the product it supports, read the information in “Notices” on page xv.

First Edition (March 2019)

This edition applies to IBM Power Systems servers that are based on the POWER9 and POWER8 processor-based technology that is running the software stack that is described in “Software levels” on page 120.

© Copyright International Business Machines Corporation 2019. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	xi
Examples	xiii
Notices	xv
Trademarks	xvi
Preface	xvii
Authors	xviii
Now you can become a published author, too!	xix
Comments welcome	xix
Stay connected to IBM Redbooks	xx
Chapter 1. Solution overview	1
1.1 Introduction	2
1.1.1 Types of AI	2
1.1.2 What is a data lake	3
1.2 Artificial intelligence solutions	4
1.2.1 IBM PowerAI	4
1.2.2 IBM Watson Machine Learning Accelerator	5
1.2.3 IBM Watson Studio Local	7
1.2.4 H2O Driverless AI	9
1.2.5 IBM PowerAI Vision	10
1.2.6 Key differences among the popular AI solutions	11
1.3 Data platforms	11
1.3.1 Apache Hadoop and Hortonworks	11
1.3.2 IBM Spectrum Scale	13
1.3.3 IBM Elastic Storage Server	15
1.3.4 IBM Spectrum Scale HDFS Transparency Connector	17
1.4 When to use a GPU	19
1.5 Hortonworks Data Platform GPU support	22
1.5.1 Native GPU support	22
1.5.2 GPU discovery	23
1.5.3 GPU isolation and monitoring	23
1.5.4 GPU scheduling	24
1.5.5 Hortonworks Data Platform 3 and YARN container with GPU support	24
1.6 Linux on Power	24
1.7 Client use cases	25
1.7.1 Asian job-hunting services company	25
1.7.2 Large bank	26
1.7.3 South American IT services provider	26
1.7.4 Governmental agency	26
1.7.5 European IT services company	26
Chapter 2. Integration overview	29
2.1 Architecture overview	30
2.1.1 Infrastructure stack	31

2.2 System configurations	33
2.2.1 IBM Watson Machine Learning Accelerator configuration	33
2.2.2 IBM Watson Studio Local configurations	35
2.2.3 Configuring an HDP system	38
2.2.4 Configuring a proof of concept	38
2.2.5 Conclusion	39
2.3 Deployment options	39
2.3.1 Deploying IBM Watson Studio Local in stand-alone mode or with IBM Watson Machine Learning Accelerator	39
2.3.2 Using the Hadoop Integration service versus using an Apache Livy connector ..	40
2.3.3 Deploying H2O Driverless AI in stand-alone mode or within IBM Watson Machine Learning Accelerator.....	40
2.3.4 Running Spark jobs.....	43
2.4 IBM Watson Machine Learning Accelerator and Hortonworks Data Platform.....	43
2.5 IBM Watson Studio Local with Hortonworks Data Platform	44
2.6 IBM Watson Studio Local with IBM Watson Machine Learning Accelerator.....	46
2.7 IBM Spectrum Scale and Hadoop Integration.....	47
2.7.1 Information Lifecycle Management.....	48
2.8 Security	50
2.8.1 Datalake security	51
2.8.2 IBM Watson Machine Learning Accelerator security with Hadoop	53
2.8.3 IBM Watson Studio Local security	53
2.8.4 IBM Spectrum Scale security	55
Chapter 3. Integrating new data.	57
3.1 Data ingestion overview	58
3.2 Types of data ingestion.....	59
3.3 Options for data ingestion	61
3.4 Using data connectors to work with external data sources	63
3.5 How integration improves the artificial intelligence models	64
Chapter 4. Integration details.	65
4.1 Integrating IBM Watson Machine Learning Accelerator with Hortonworks	66
4.1.1 Running remote Spark jobs with Livy	67
4.1.2 Accessing the Hadoop data from IBM Watson Machine Learning Accelerator..	69
4.2 Integrating IBM Watson Studio Local, IBM Watson Machine Learning Accelerator, and Hadoop clusters	72
4.3 Integrating IBM Watson Studio Local with Hortonworks Data Platform	79
Chapter 5. Accessing real-time data	87
5.1 Hortonworks DataFlow	88
5.1.1 Planning a Hortonworks DataFlow installation	89
5.2 Apache NiFi.....	90
5.2.1 Adding the Apache NiFi service to an HDF cluster.....	91
5.2.2 Working with Apache NiFi.....	91
5.2.3 Integrating Apache NiFi with a data science tool environment	92
5.3 Apache Storm	92
5.3.1 Working with Apache Storm	93
5.3.2 Integrating Apache Storm with a data science tool environment	93
5.4 Apache Spark Streaming	94
5.4.1 Working with Apache Spark Streaming.....	95
5.5 Apache Kafka Streams	97
5.6 Integrating streaming tools with data science tools.....	100
5.6.1 Application overview	100

5.6.2 Configuring a Kafka topic	101
5.6.3 Starting a streamer by using the nmap-ncat utility	102
5.6.4 Running the Spark Streaming engine.....	103
5.6.5 Merging data by using NiFi and saving it on HDFS	105
5.6.6 Conclusion of the data stream integration proof of concept	116
Appendix A. Additional information	119
System topology.....	120
Software levels.....	120
Appendix B. Installing an IBM Watson Machine Learning Accelerator notebook ..	121
Customizing a notebook package	122
Adding a notebook package.....	123
Creating a Spark Instance Group with a notebook	125
Creating notebooks for users.....	130
Testing notebooks	133
Conclusion and additional information	134
Related publications	135
IBM Redbooks	135
Online resources	135
Help from IBM	136

Figures

1-1 A diagram showing how different computer models relate to each other	2
1-2 Performance relative to DL and ML	3
1-3 IBM PowerAI distribution	5
1-4 IBM Watson Machine Learning Accelerator parts	6
1-5 IBM Watson Machine Learning Accelerator supported frameworks and libraries	7
1-6 IBM Watson Studio Local components	8
1-7 IBM Watson Studio Local node roles	9
1-8 H2O Driverless AI console	10
1-9 IBM PowerAI Vision	10
1-10 Key differences among the AI solutions for Power Systems servers	11
1-11 Applications interacting with data systems and sources	12
1-12 Hortonworks software stack on top of the IBM Spectrum file system	13
1-13 IBM Spectrum Scale	15
1-14 IBM ESS hardware and software stack configuration	17
1-15 Overall dataflow between IBM Spectrum Scale, storage, and the compute nodes	18
1-16 NVIDIA Volta GPU specifications	21
1-17 The nvidia-smi command output	23
2-1 How the components fit together	30
2-2 Disk space of a VM	36
2-3 CPU output of a VM	36
2-4 HDP recommended configurations	38
2-5 H2O Driverless AI as shown in the management console	41
2-6 Watson Studio connections with Hortonworks	45
2-7 IBM Spectrum Scale flow of data architecture	47
2-8 IBM Spectrum Scale GUI: Information Lifecycle Add Rule window	49
2-9 IBM Spectrum Scale GUI: Information Lifecycle Rule Type window	49
2-10 IBM Spectrum Scale: Information lifecycle rules configuration	50
2-11 IBM Watson Studio Local Hadoop registration service	54
3-1 Data ingestion	58
3-2 Hortonworks Data Platform	59
3-3 Hortonworks Data Platform overview	60
3-4 HDF data ingestion	60
3-5 Apache NiFi	61
3-6 Deep learning	62
3-7 All data sets	63
3-8 Application layer view	63
3-9 Types of data connectors	64
4-1 IBM Watson Machine Learning Accelerator with Hortonworks and Livy	67
4-2 Hadoop data access from IBM Watson Machine Learning Accelerator	69
4-3 Adding a data connector in an IBM Watson Machine Learning Accelerator SIG	70
4-4 New Data Connector details	71
4-5 IBM Watson Studio Local with IBM Watson Machine Learning Accelerator	72
4-6 Registering a new application instance	73
4-7 Specifying the application template	73
4-8 Specifying an application name	73
4-9 Specifying the top-level consumer for the application	74
4-10 Selecting the resource groups	74
4-11 Specifying the repository packages	74

4-12 Specifying the parameter values associated with the SIG	75
4-13 The Livy2 application in the Registered state	75
4-14 The started Livy2 Application showing the URL in the outputs	76
4-15 The IBM Watson Studio Local to IBM Watson Machine Learning Accelerator Livy Notebook Test	77
4-16 IBM Watson Studio Local with Hortonworks	79
4-17 Add Registration window	80
4-18 Starting the environment	82
4-19 Installing the package	83
4-20 Saving the image	83
4-21 Selecting Hadoop Integration from the admin console	84
4-22 Displaying the details of the registered Hadoop system	84
4-23 Pushing the run time	85
5-1 Hortonworks Data Platform components	88
5-2 Hortonworks DataFlow components	89
5-3 Data flow on the NiFi GUI	91
5-4 Basic Storm application topology	93
5-5 Flow of the Spark Streaming engine	95
5-6 Apache Kafka architecture	98
5-7 Anatomy of a Kafka topic	98
5-8 The anatomy of a Kafka topic partition	99
5-9 Kafka Streams integration into a Kafka flow	99
5-10 Kafka Streams components and application flow	100
5-11 Proof of concept application overview	101
5-12 Spark History Server	105
5-13 NiFi GUI: Adding a processor to the project	106
5-14 NiFi GUI: Selecting the GetKafka processor	107
5-15 NiFi GUI: Selecting the MergeContent processor	108
5-16 NiFi GUI: Selecting the PutHDFS processor	109
5-17 NiFi GUI: Application overview	109
5-18 NiFi GUI: Linked processors	110
5-19 NiFi GUI: GetKafka processor properties	111
5-20 NiFi GUI: MergeContent processor properties	111
5-21 NiFi GUI: PutHDFS processor properties	112
5-22 NiFi GUI: MergeContent processor settings	113
5-23 NiFi GUI: PutHDFS processor settings	114
5-24 NiFi GUI: Starting the first data flow processor	115
5-25 NiFi GUI: Checking queued messages	115
5-26 NiFi GUI: Merged file on the queue between the MergeContent and PutHDFS processors	116
A-1 System topology	120
B-1 Notebook Management menu	123
B-2 Notebook Management user interface	123
B-3 Add Notebook wizard	124
B-4 Notebook copy in progress	125
B-5 Spark Instance Groups menu	125
B-6 Creating a Spark Instance Group	126
B-7 New Spark Instance Group: Part 1	126
B-8 New Spark Instance Group: Part 2	127
B-9 New Spark Instance Group: Part 3	127
B-10 New Spark Instance Group: Part 4	128
B-11 Starting a Spark Instance Group	129
B-12 The Creating Notebooks for Users menu option	130

B-13	Creating Notebooks for Users details	131
B-14	My Notebooks window	132
B-15	Notebook server login.	133
B-16	Notebook server home page	133
B-17	Notebook test	134

Tables

1-1 Concepts that define a data lake	3
2-1 Software levels	30
2-2 System details.....	31
2-3 IBM ESS GS2	32
2-4 Hardware requirements	34
2-5 Software requirements	34
2-6 Three-node configuration	35
2-7 Sample configuration of a 7-node cluster.....	37
2-8 Nine-node configuration	37
2-9 Proof of concept configuration	38
A-1 Software levels recommendations.....	120

Examples

4-1	Code snippet to perform logistic regression by using Sparkmagic	68
4-2	Code snippet to perform logistic regression by using remote data	69
4-3	Loading data by using an HDFS connector	71
4-4	Running the TensorFlow Keras model	77
4-5	Notebook code to get the list of Hadoop clusters	80
4-6	List of available Hadoop systems	80
4-7	Configuring a Spark session	81
4-8	Setting up Sparkmagic	81
4-9	Starting the remote Livy session	81
4-10	Running Spark on the Hadoop cluster from the IBM Watson Studio Local notebook .	81
5-1	Python code: Importing and instantiating SparkContext and Streaming Context	96
5-2	Python code: Creating a DStream	96
5-3	Python code: Splitting the lines into words	96
5-4	Python code: Counting the words	96
5-5	Python code: Starting the StreamingContext .start() function	97
5-6	Program output	97
5-7	Linux terminal: Creating a Kafka topic	101
5-8	Linux terminal: Listing the Kafka topic information	101
5-9	Linux terminal: Producing messages on the Kafka topic	102
5-10	Linux terminal: Consuming messages on a Kafka topic	102
5-11	Linux terminal: yum	102
5-12	Linux terminal: Starting nc at port 9999 with the output of /var/log/messages .	102
5-13	Linux terminal: Listing the active TCP ports	103
5-14	Linux terminal: Testing the TCP service with telnet	103
5-15	Python code: Declaring libraries	103
5-16	Python code: Defining a function	104
5-17	Python code: Defining Spark objects	104
5-18	Python code: Defining the DStream	104
5-19	Python code: Running the send_to_kafka function	104
5-20	Python code: Starting the Spark Streaming job	105
5-21	Linux Terminal: Submitting a Spark job by using a Python program	105
5-22	Linux terminal: Listing data that is stored on HDFS	116

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®	IBM Spectrum Conductor™	POWER9™
Cognos®	IBM Spectrum Scale™	Redbooks®
Db2®	IBM Watson®	Redbooks (logo)  ®
GPFS™	OpenCAPI™	SPSS®
IBM®	POWER®	Tivoli®
IBM Cloud™	Power Architecture®	Watson™
IBM Elastic Storage™	Power Systems™	Watson Health™
IBM Spectrum™	POWER8®	WebSphere®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

As big data becomes more ubiquitous, businesses are wondering how they can best leverage it to gain insight into their most important business questions. Using machine learning (ML) and deep learning (DL) in big data environments can identify historical patterns and build artificial intelligence (AI) models that can help businesses to improve customer experience, add services and offerings, identify new revenue streams or lines of business (LOBs), and optimize business or manufacturing operations. The power of AI for predictive analytics is being harnessed across all industries, so it is important that businesses familiarize themselves with all of the tools and techniques that are available for integration with their data lake environments.

In this IBM® Redbooks® publication, we cover the best practices for deploying and integrating some of the best AI solutions on the market, including:

- ▶ IBM Watson® Machine Learning Accelerator (see note for product naming)
- ▶ IBM Watson Studio Local
- ▶ IBM Power Systems™
- ▶ IBM Spectrum™ Scale
- ▶ IBM Data Science Experience (IBM DSX)
- ▶ IBM Elastic Storage™ Server
- ▶ Hortonworks Data Platform (HDP)
- ▶ Hortonworks DataFlow (HDF)
- ▶ H2O Driverless AI

We map out all the integrations that are possible with our different AI solutions and how they can integrate with your existing or new data lake. We also walk you through some of our client use cases and show you how some of the industry leaders are using Hortonworks, IBM PowerAI, and IBM Watson Studio Local to drive decision making. We also advise you on your deployment options, when to use a GPU, and why you should use the IBM Elastic Storage Server (IBM ESS) to improve storage management. Lastly, we describe how to integrate IBM Watson Machine Learning Accelerator and Hortonworks with or without IBM Watson Studio Local, how to access real-time data, and security.

Note: IBM Watson Machine Learning Accelerator is the new product name for IBM PowerAI Enterprise.

Note: Hortonworks merged with Cloudera in January 2019. The new company is called Cloudera. References to *Hortonworks* as a business entity in this publication are now referring to the merged company. Product names beginning with Hortonworks continue to be marketed and sold under their original names.

Authors

This book was produced by a team of specialists from around the world working at IBM Redbooks, Austin Center.

Ivaylo B. Bozhinov has been working at IBM Bulgaria for 4 years as a technical support professional. His main areas of expertise are IBM Power Systems products, IBM AIX®, IBM i, and Red Hat Enterprise Linux. He has a bachelor's degree in Information Technology from the State University of Librarian and Information Technology of Sofia, Bulgaria. He holds several IBM certifications, which include Hadoop Administration, Hadoop Foundation, Data Science Foundation, IBM Private Cloud, and IBM Blockchain Foundation. His areas of interest include AI, DL, ML, blockchain, and cloud.

Anto Ajay Raj John is a Senior Software Architect working at IBM India for the past 13 years. He works as a Performance Analyst for Deep Learning (AI) frameworks on IBM POWER® processor-based platforms. He is responsible for design input for future IBM enterprise AI Systems and optimizing the stack. Before he worked on DL, he worked in areas like high-performance computing (HPC), system solving, and enterprise operating systems. His areas of interest include AI, distributed systems, and future systems. He holds a few patents and publications and a Master of Engineering degree in Computer Science from BITS Pilani, India.

Rafael Freitas de Lima is a Solution Architect at IBM Brazil for analytics and cognitive solutions. Since he joined IBM in 2009, he works with data analytics, and for the last 3 years with cloud-native applications and Watson services. He holds a BSc degree in Information Systems and an MBA in Big Data. His areas of expertise include Business Intelligence (BI), cloud, conversational systems, and big data.

Ahmed (Mash) Mashhour is a Power Systems Global subject matter expert (SME). He is IBM AIX, Linux, and IBM Tivoli® certified with 14 years of professional experience in IBM AIX and Linux systems. He is an IBM AIX back-end SME who supports several customers in the US, Europe, and the Middle East. His core experience is in IBM AIX, Linux systems, clustering management, virtualization tools, and various Tivoli and database products. Mash has authored several publications inside and outside IBM. He has hosted more than 70 classes around the world.

James Van Oosten works in IBM Lab Services with a focus on AI solutions on Power Systems servers. He is an expert on ML and DL. His prior roles include analytic subsystem development in IBM Watson Health™ Cloud, control system development on the Blue Gene Supercomputer, enterprise Java development in IBM WebSphere® Application Server, and storage management development on IBM i systems.

Fernando Vermelho is an IBM System Lab Services Senior Big Data and Deep Learning Specialist in Brazil. He has worked at IBM since 2008 when he joined as a Client Technical Specialist for AS400. He moved to AIX and Power Systems servers. He holds a degree in IT Management from the Universidade Paulista - Brazil and an extension of Big Data Analysis through Machine Learning from Faculdade Instituto de Administração (FIA) in Brazil. His areas of expertise include ML with large data sets, DL, IBM Watson Visual Recognition patterns, AIX, and Power Systems servers.

Allison White is a Customer Success Manager and Sales Representative who works primarily in Enterprise Software startups. She has a Master of Science degree in Management with a focus on Digital Business from the Nova School of Business and Economics in Lisbon, Portugal, and a Bachelor of Arts degree from the University of Texas at Austin, Texas, US. Her main areas of interest are customer success, technical sales, and product marketing.

The project that produced this publication was managed by
Scott Vetter, PMP

Thanks to the following people for their contributions to this project:

Jesus Alvarez, Kathy Bennett, Jeffery Bisti, Army Brown, Jason M. Furmanek, Lotic Fura, Diane Fahr, Marty Fullam, Geoffery Goldman, Jim Gosack, Beth Hoffman, Kristy Knight, Rajyalakshmi D Marathu, Wanye Namerow, Kanchana Padmanabhan, Indajilt Poddar, Keshav Ranganathan, Steve Roberts, Akhil Tandon, Dustin VanStee, James Wang, Jim Woodbury, Theresa Xu, and Steve Zehner

IBM

Zohar Nissare-Houssen
Cloudera

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Solution overview

This chapter provides an introduction to the technologies that play a key part in the integration of big data and artificial intelligence (AI).

The following topics are covered in this chapter:

- ▶ Introduction
- ▶ Artificial intelligence solutions
- ▶ Data platforms
- ▶ When to use a GPU
- ▶ Hortonworks Data Platform GPU support
- ▶ Linux on Power
- ▶ Client use cases

1.1 Introduction

This publication is focused on the best practices for integrating IBM AI solutions, including IBM PowerAI, IBM Watson Machine Learning Accelerator, and IBM Watson Studio Local with big data solutions like Hortonworks Data Platform (HDP) and Hortonworks DataFlow (HDF). Integration of H2O Driverless AI is also covered.

Note: IBM Watson Machine Learning Accelerator is the new product name for IBM PowerAI Enterprise.

1.1.1 Types of AI

A review of some of the terminology in Figure 1-1 might be required before you learn about the solutions.

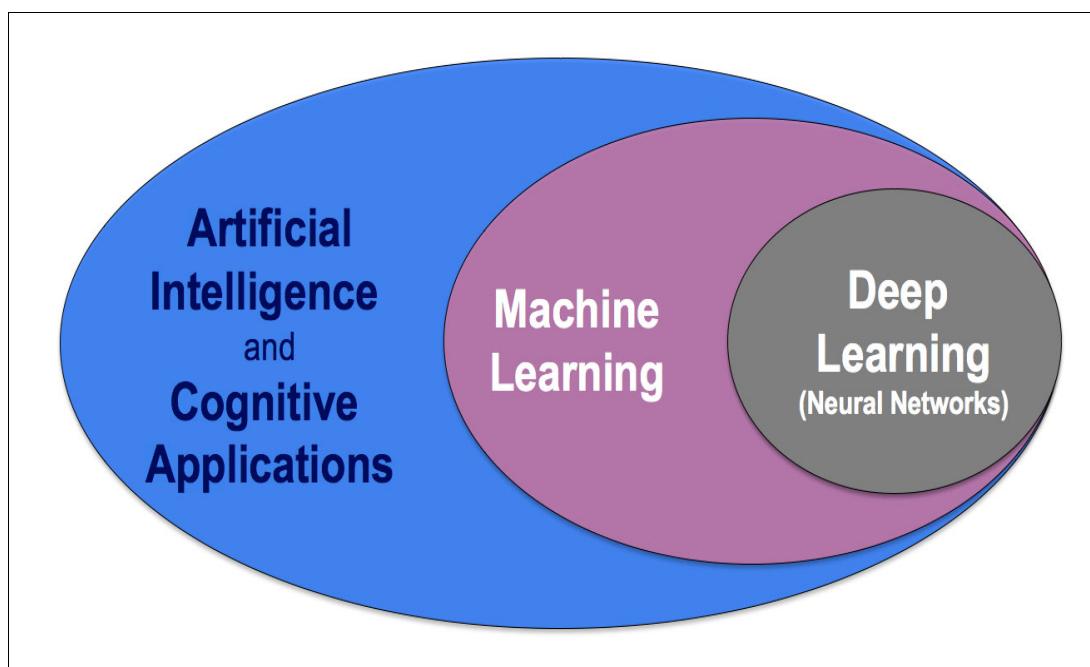


Figure 1-1 A diagram showing how different computer models relate to each other

AI is any technique that enables computers to mimic human intelligence, which includes the general form of AI that you see in science fiction movies. Cognitive applications work with humans to extend and augment their areas of expertise.

Machine learning (ML) is an application of AI that can learn and improve from experience without being explicitly programmed. ML includes supervised, unsupervised, and reinforced learning:

- ▶ *Supervised learning* uses data that has both the input and the output to form a mathematical model that can be used to make an output prediction on new input data.
- ▶ *Unsupervised learning* takes a set of input data and looks for groupings or classes within the data.
- ▶ *Reinforced learning* is an area of ML where an agent learns by being rewarded or not based on actions that are taken. It is used in game playing and autonomous vehicles.

Deep learning (DL) is a subset of ML that uses layers of artificial neural nets, a type of processing unit, to extract features from vast amounts of data to get to the answers. DL models can be trained in a supervised fashion for problems like classification of images or in an unsupervised fashion for problems like pattern analysis.

Figure 1-2 shows that classical ML techniques can outperform DL when training with small to medium amounts of data, but they reach a limit where the models cannot improve learning with more data. DL technologies excel with vast amounts of data because DL models can be extended by adding layers so that they can represent and interpret more information from more data. Data can be thought of as the fuel for DL, and Apache Hadoop data lakes can be an excellent source of that fuel.

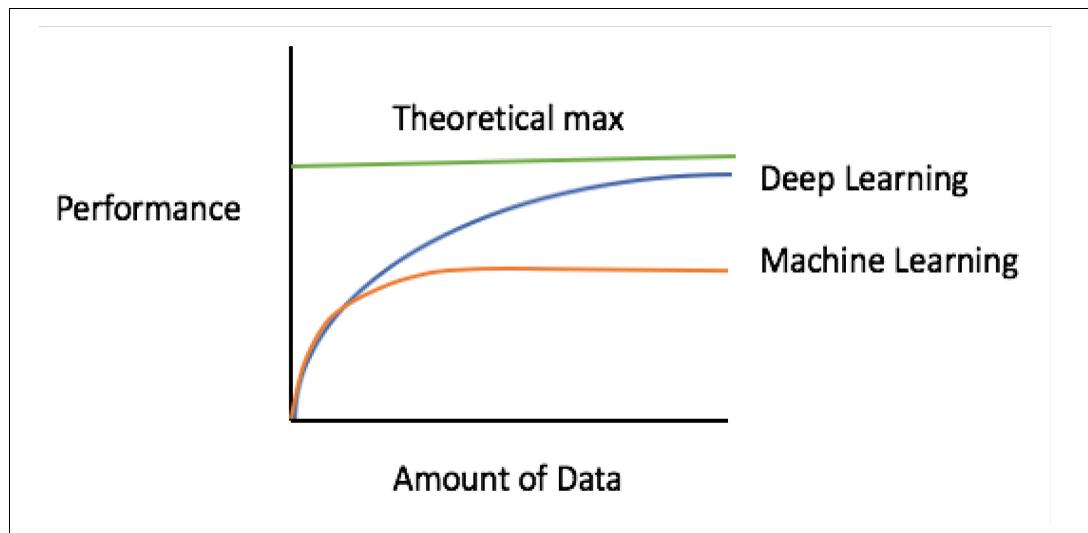


Figure 1-2 Performance relative to DL and ML

1.1.2 What is a data lake

There are numerous views out there about what constitutes a data lake, many of which are overly simplistic. At its core, a data lake is a central location in which to store all your data regardless of its source or format. It is typically built by using Hadoop or another scale-out architecture that can store cost-effectively significant volumes of data.

One of most common misunderstandings is confusing the concepts of a data lake and data warehouse. Table 1-1 provides a comparison of the key attributes of a data warehouse in contrast to a data lake.

Table 1-1 Concepts that define a data lake

Attribute	Data warehouse	Data lake
Schema	Schema-on-write.	Schema-on-read.
Scale	Scales to moderate to large volumes at moderate cost.	Scales to huge volumes at low cost.
Access Methods	Accessed through standardized SQL and Business Intelligence (BI) tools.	Accessed through SQL-like systems and programs that are created by developers. Also supports big data analytics tools.

Attribute	Data warehouse	Data lake
Workload	Supports batch processing and thousands of concurrent users performing interactive analytics.	Supports batch and stream processing, and has an improved capability over data warehouses to support big data inquiries from users
Data	Cleansed.	Raw and refined.
Data Complexity	Complex integrations.	Complex processing.
Cost/Efficiency	Efficiently uses CPU/IO but has high storage and processing costs.	Efficiently uses storage and processing capabilities at low cost.

Because all data can be stored in them, data lakes are a powerful alternative to the challenges that are presented by data integration in a traditional data warehouse, especially as organizations turn to mobile and cloud-based applications and the Internet of Things (IoT). Companies also want to know how to train, classify at scale, and set up DL pipelines while using the existing Hadoop data lake and the data and processing power.

1.2 Artificial intelligence solutions

This section describes the AI solutions for IBM Power Systems servers in more detail. It provides a summary of the functional differences of the solutions.

1.2.1 IBM PowerAI

In this section, we introduce IBM PowerAI (also referred to as IBM PowerAI base) and IBM Watson Machine Learning Accelerator.

IBM PowerAI base

IBM PowerAI base is a downloadable enterprise software distribution that includes open source ML frameworks like scikit-learn and DL frameworks like TensorFlow, PyTorch, and Caffe. It is available at no charge. It is hardened and supported for Power Systems servers. IBM PowerAI integrated with IBM Watson Studio Local is one of the available notebook environments. IBM PowerAI is also included in IBM Watson Machine Learning Accelerator for large-scale deployments.

IBM PowerAI includes the following features to boost performance and reduce training time:

- ▶ A distributed deep learning (DDL) library that scales out DL to hundreds of Power Systems servers and leverages the advantages of IBM POWER9™ processor-based servers and NVIDIA GPUs. DDL is limited to four nodes on IBM PowerAI, so use IBM Watson Machine Learning Accelerator for larger configurations.
- ▶ A Snap ML library that brings distributed GPU acceleration to ML algorithms like scikit-learn.

- ▶ Large model support (LMS) that enables models to be larger (that is, include higher resolution images) by integrating with DL frameworks. LMS uses the fast NVLinks on the POWER9 processors to swap portions of the model to and from main memory at various points during training. TensorFlow LMS performance is improved in IBM PowerAI V1.5.4, and LMS for PyTorch is provided as a technology preview.
- ▶ The frameworks are built to leverage the advantages of IBM POWER9 processor-based servers and NVIDIA GPUs.

Figure 1-3 provides a stack view of the components that make up IBM PowerAI.

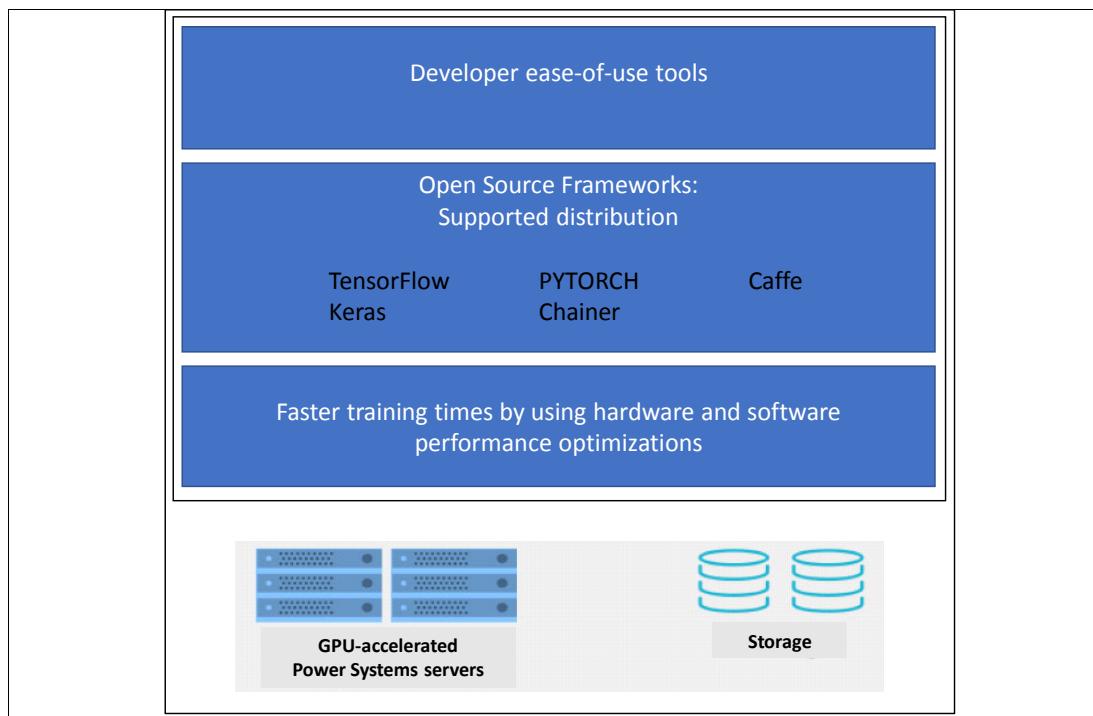


Figure 1-3 IBM PowerAI distribution

1.2.2 IBM Watson Machine Learning Accelerator

IBM Watson Machine Learning Accelerator is an enterprise-ready AI solution that includes IBM PowerAI and IBM Spectrum Conductor™. It provides an end-to-end DL platform for groups of data scientists. It can create and manage Spark clusters that are called Spark Instance Groups (SIGs), which share the underlying resources of the IBM Watson Machine Learning Accelerator servers, including memory, CPUs, and GPUs. The resource sharing is controlled by the scheduling policy, which can be adjusted dynamically to accelerate job completion.

Note: In this publication, IBM Watson Machine Learning Accelerator is the new product name for IBM PowerAI Enterprise.

IBM Watson Machine Learning Accelerator includes DDL and LMS for faster time to results. The model development tools include real-time training visualization, accuracy monitoring at run time, and hyper-parameter search and optimization. The DDL can scale to thousands of nodes.

IBM Watson Machine Learning Accelerator combines popular open source DL frameworks, efficient AI development tools, and accelerated Power Systems servers. Now, organizations can deploy a fully optimized and supported AI platform that delivers blazing performance, dependability, and resilience. IBM Watson Machine Learning Accelerator is a complete environment for data science as a service (DSaaS), enabling your organization to bring new applied AI applications into production.

The IBM Watson Machine Learning Accelerator software bundle includes Anaconda sourced libraries, IBM PowerAI base, and IBM Spectrum Conductor Deep Learning Impact.

Figure 1-4 provides the major components of IBM Watson Machine Learning Accelerator.

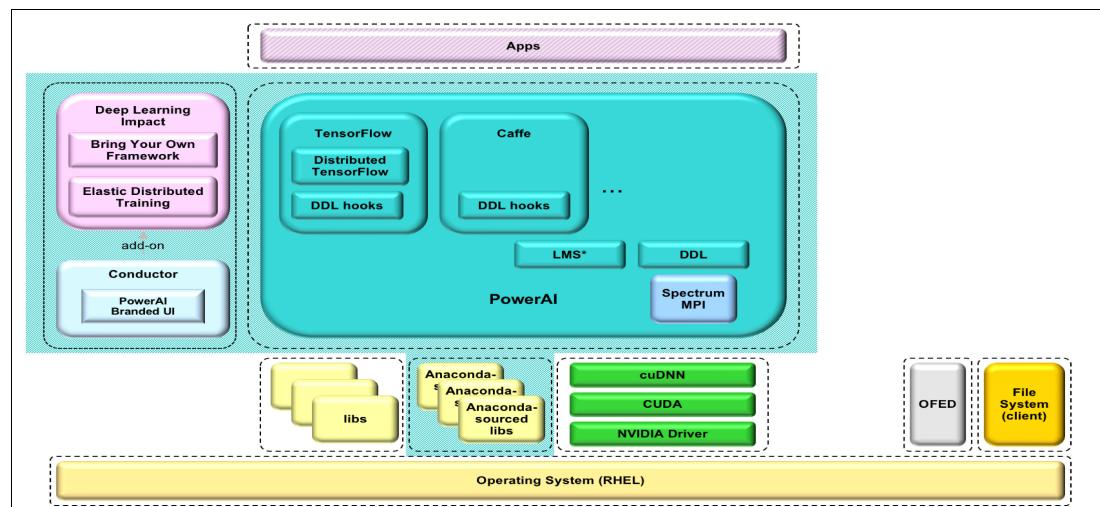


Figure 1-4 IBM Watson Machine Learning Accelerator parts

Anaconda is an open source distribution of Python and R programming languages for data science and ML that simplifies the management and deployment of packages. IBM Spectrum Conductor provides multitenancy by representing lines of business (LOBs) as consumers, each with their own set of SIGs that can share resource pools and be controlled by dynamic policies. IBM Spectrum Conductor Deep Learning Impact includes auto hyper-parameter tuning that can create tens of training jobs with different hyper-parameters, and then monitor the jobs and remove ones that appear less promising.

Figure 1-5 shows the supported frameworks and libraries that make up the IBM Watson Machine Learning Accelerator Platform.

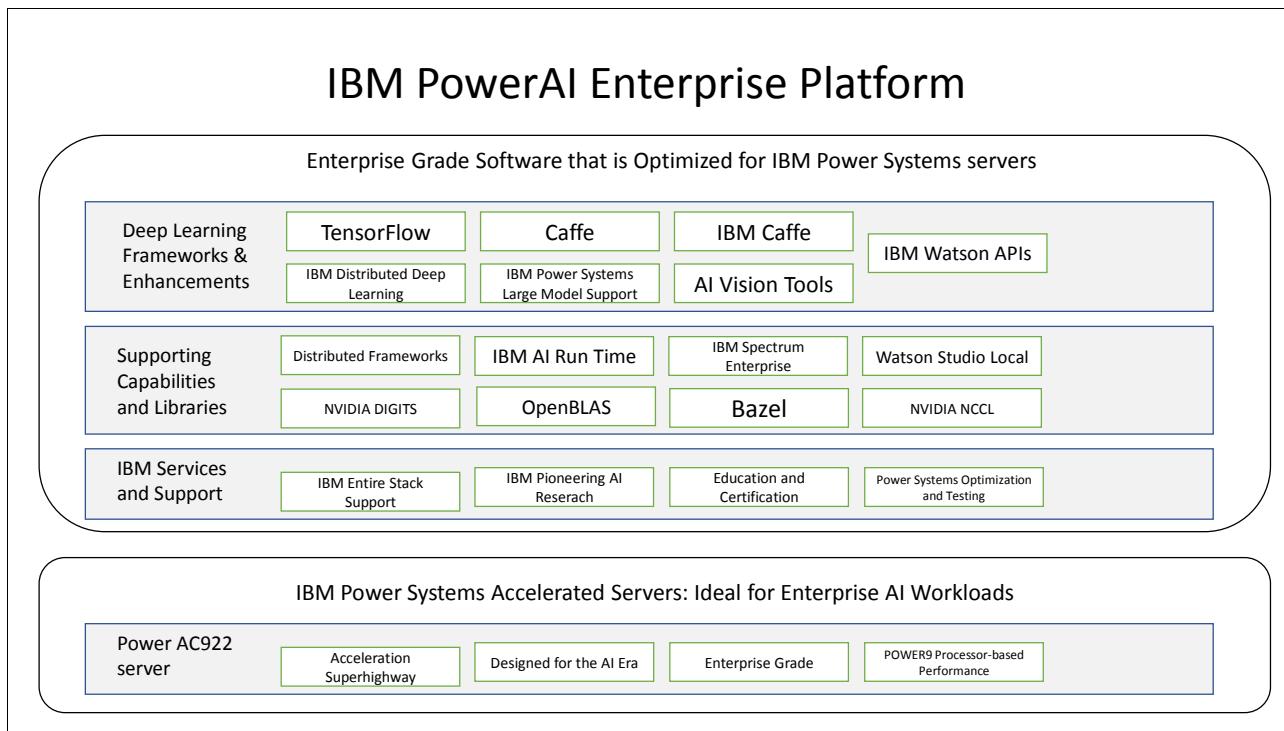


Figure 1-5 IBM Watson Machine Learning Accelerator supported frameworks and libraries

1.2.3 IBM Watson Studio Local

IBM Watson Studio Local (formerly called IBM Data Science Experience (IBM DSX) Local) is an enterprise solution for data scientists and data engineers. It includes IBM PowerAI and a suite of data science tools like RStudio, Spark, Jupyter Notebooks, and Zeppelin Notebooks.

Work is organized in projects that consist of notebooks, models, data sources, and remote data assets. Users associate their tools like Jupyter Notebooks with R, Python, and Scala kernels with Spark runtime environments. The environments can be customized and saved. Data assets include files and connections to existing data sources like Hadoop Distributed File Systems (HDFSs) and databases. Users can transform and shape the data by using the Data Refinery component.

IBM Watson Studio Local is the on-premises solution version of the IBM Watson Studio product. It is a platform that provides tools like RStudio, Spark, Jupyter Notebooks, and Zeppelin Notebooks for data scientists, data engineers, application developers, and subject matter experts (SMEs) to work collaboratively and easily with data to build and train models at scale. It gives the flexibility to build models where your data is and deploy them anywhere in a hybrid environment.

Figure 1-6 shows the major components of IBM Watson Studio Local.

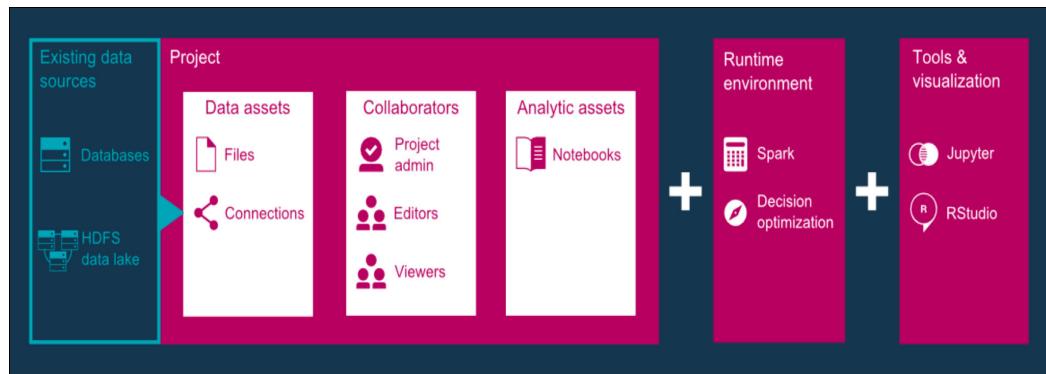


Figure 1-6 IBM Watson Studio Local components

There are several roles that are supported for project collaborators, including project administrators, editors, and viewers. A community section with sample notebooks is provided to help accelerate developer time to value. An administration console is provided to manage and monitor hardware, users, and services.

IBM Watson Studio Local runs on a Kubernetes cluster of servers that take on roles of master (control plane), storage, and compute. The architecture requires a minimum of four nodes. The deployment role is optional when you do the installation, but is required if models are deployed. IBM Watson Studio Local can manage its users for authentication or integrate with an existing Lightweight Directory Access Protocol (LDAP) enterprise directory server.

Figure 1-7 is a graphical representation of the roles of the various nodes in IBM Watson Studio Local.

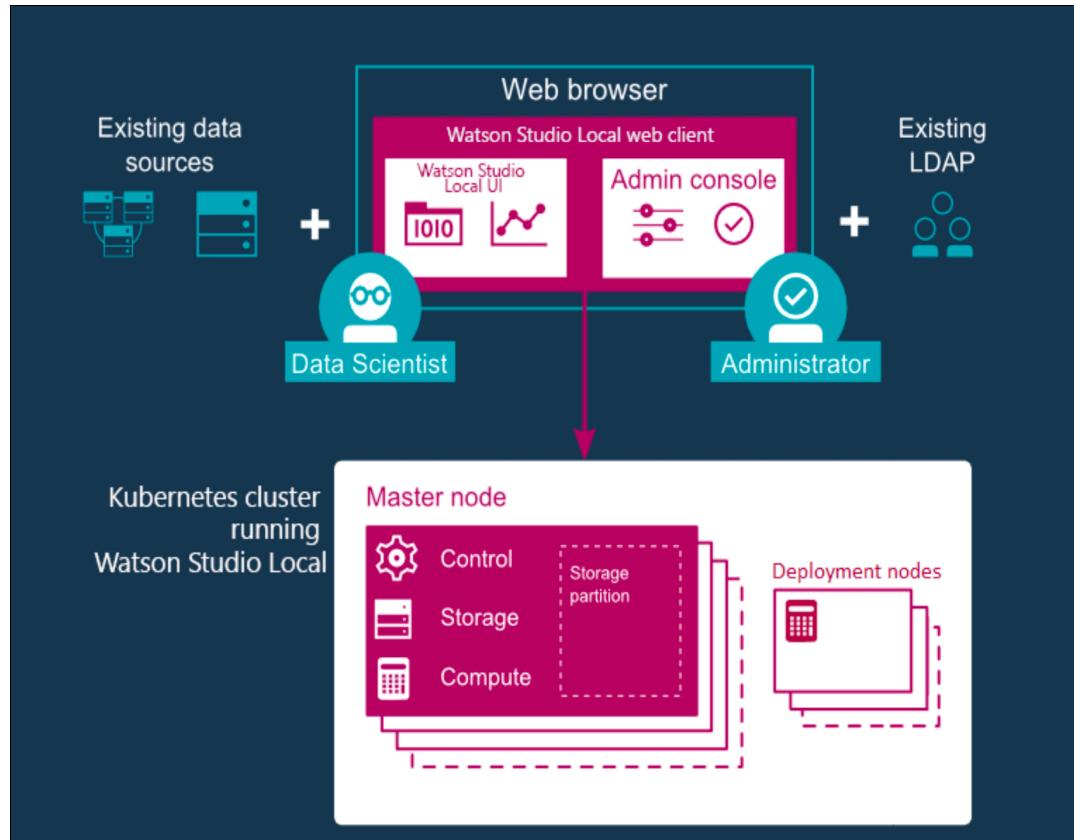


Figure 1-7 IBM Watson Studio Local node roles

1.2.4 H2O Driverless AI

H2O Driverless AI is H2O.ai's solution for automated machine learning (ML). It simplifies many of the data science ML tasks. Data can be ingested from the cloud, Hadoop, or desktop systems. The data visualization is automatic, showing the data shape, outliers, and missing values. The automated ML uses best practice model recipes and the underlying machine power to iterate across thousands of possible models, which includes feature engineering and parameter tuning. The automated scoring pipeline includes the feature transformations and models for rapid deployment to production.

H2O Driverless AI is supported on IBM Power Servers, leverages GPUs for accelerated ML, and can be installed and managed as an WLM-A application.

Figure 1-8 shows a sample of the H2O Driverless AI console.



Figure 1-8 H2O Driverless AI console

1.2.5 IBM PowerAI Vision

IBM PowerAI Vision eases the task of labeling objects in images and video through a “point and click” interface that enables non-SMEs to contribute to the data science project.

The various stages of ML are shown in Figure 1-9.

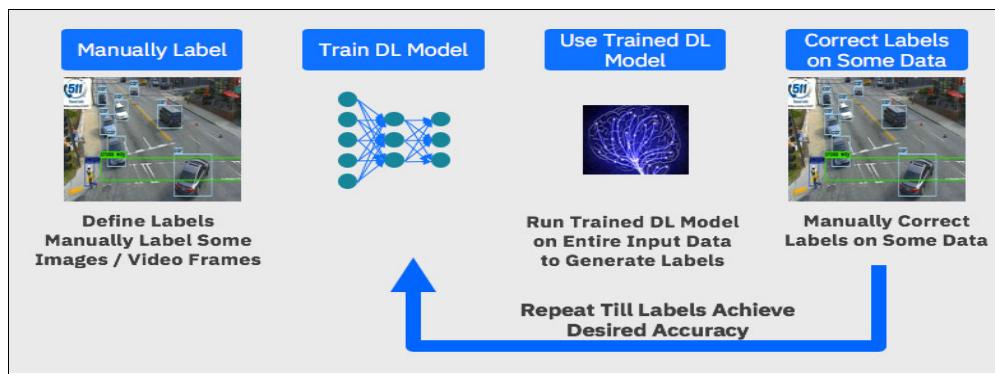


Figure 1-9 IBM PowerAI Vision

A model can be trained with an initial set of labeled data and used to label new images. The new labeled images can be reviewed, corrected, and used to update the model to improve its accuracy. The process is repeated until the necessary model accuracy is reached. Using IBM PowerAI Vision is beyond the scope of this book.

1.2.6 Key differences among the popular AI solutions

Figure 1-10 shows the key differences of the solutions. Both IBM PowerAI and IBM Watson Machine Learning Accelerator support DDL, but IBM Watson Machine Learning Accelerator can scale to thousands of nodes. The IBM Watson Studio Local notebook-based environment is a good choice for data science teams that are comfortable working with Jupyter Notebooks and R. H2O Driverless AI and AI Vision enable people with lower skills to work on AI projects.

	Power AI Base	Power AI Enterprise	AI Vision	ML and DL Watson Studio Local	Machine Learning H2O Driverless AI
Offering	Deep Learning	Deep Learning for the Enterprise	Deep Learning with Video tools	Notebook oriented development environment for ML and DL	Automated Machine learning
Applications	Text & Numeric Yes Images Yes Video -	Yes Optional add-on Data Scientist IT	No Yes Yes	Yes Yes	Yes No No
Primary Persona	Data Scientist IT	Data Scientist IT	Line of Business IT	Data Scientist IT	Data Scientist Line of Business
Second persona					
User Skill Level	High	Medium to high	Low	Medium to high	Low to Medium
Strengths	Rapid deployment, high performance, scale	enterprise grade, High performance, rapid Deployment	Rapid deployment, simple GUI high performance	Notebook based development environment, strong collaboration, model management	Simplified deployment, intuitive user interface, automatic pipelines, "explainability" for models, end to end automation
Platform	1-4 nodes	1-thousands of nodes	-	-	-
Large Model Support Server(s)	Yes	Yes	-	-	-
IBM Products	S822LC or AC922	S822LC or AC922	S822LC or AC922	S822LC or AC922, LC922	S822LC, AC922, LC921/922
Spectrum MPI (DDL)	Limited to 4 nodes	Included	-		Optional add-on
Spectrum Conductor DLI	Optional add-on	Included	-	Optional Add On	Optional add-on
IBM Watson Studio Local	Optional add-on	Optional add-on	No		Optional add-on
Cloud	IBM Cloud Public Yes	No	Trial only	Watson Studio	-
	IBM Cloud Private Yes	Yes	Yes	Yes	-

Figure 1-10 Key differences among the AI solutions for Power Systems servers

1.3 Data platforms

In this section, we introduce the data platforms on which this publication is based.

1.3.1 Apache Hadoop and Hortonworks

Apache Hadoop is one of the most popular data lake technologies. It is a highly scalable open source platform that can process very large data sets across hundreds to thousands of parallel computing nodes. It provides a cost-effective storage solution for data ingestion with no initial format requirements. As an open source platform, it is built on code contributions from a community of some of the world's best developers.

Figure 1-11 shows the relationship between applications, the databases, and the sources of that data.

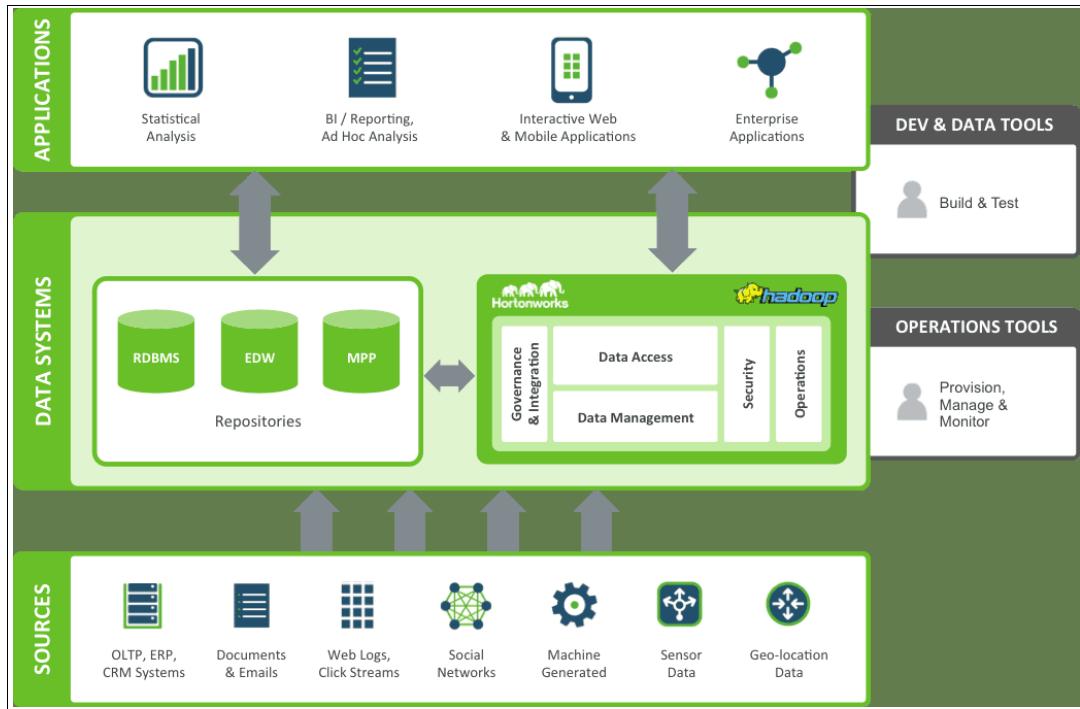


Figure 1-11 Applications interacting with data systems and sources

Hortonworks Data Platform

HDP is a Hadoop and Spark distribution that enables enterprises to store, process, and analyze securely large amounts of data-at-rest. It includes an HDFS that can scale to thousands of nodes.

The storage for HDFS can be mapped to local drives or to IBM Elastic Storage Server (IBM ESS) through an IBM Spectrum Scale™ HDFS Transparency connector. HDP includes Apache Spark, which is a distributed processing framework that includes MLlib, which is the Spark ML library.

With the emergence of new business models and innovations, Hadoop is at the verge of reinventing itself to be the next-generation data platform that powers technology advancement. With a newer, more powerful version release of Hadoop in 2018, the partnership of IBM and Hortonworks extended its capabilities, driving new levels of data science and ML, and further building Hadoop into an enterprise-class data platform with advanced analytic capabilities. The result is an integrated solution that combines HDP, HDF, and IBM technology.

This IBM and Hortonworks solution enables a data-lake-based Hadoop infrastructure with improved exploration, discovery, testing, and advanced data query. It also offers massive scalability, security, and governance with the ability to federate both data-at-rest and data-in-motion across the entire organization. Users can easily query both relational databases and Hadoop on-premises or in the cloud. They benefit from self-service data access and the ability to do ad hoc and real-time queries.

HDP is the data lake of choice for IBM Power Systems servers, and it integrates well with IBM Watson Studio Local.

Hortonworks DataFlow

HDF provides real-time streaming analytics for data-in-motion that complements HDP. It is an integrated solution that has Apache Nifi and MiNifi, Kafka, Storm, Streaming Analytics Manager, and Schema Registry to provide a real-time streaming data analytics platform. It is managed by an Ambari installation and coordinated by Zookeeper.

1.3.2 IBM Spectrum Scale

IBM Spectrum Scale (formerly IBM General Parallel File System (IBM GPFS™)) is a cluster file system that provides concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be SAN-attached, network-attached, a mixture of SAN-attached and network-attached, or in a shared-nothing cluster configuration. These setups enable high-performance access to this common set of data to support a scale-out solution or to provide a high availability platform. IBM Spectrum Scale is the parallel file system at the heart of IBM ESS.

IBM Spectrum Scale enables the unification of virtualization, analytics, and file and object use cases into a single scale-out storage solution, and it provides a single namespace for all data, which offers a single point of management. With the support of a wide range of multiple protocols of communication for data, such as POSIX file systems, Network File System (NFS), Server Message Block (SMB), Object Storage like Swift and S3, and block storage like iSCSI and HDFS, customers can run their in-place analytics workloads without needing to duplicate data sets.

These various communication protocols offer several benefits, such as consolidating various data sources efficiently in one global namespace. IBM Spectrum Scale provides a unified data management solution, enables efficient space utilization, and avoids unnecessary data moves because the access methods might be different.

Using storage policies transparent to users, data can be compressed or tiered to help cut costs. Data can also be tiered to high-performance media, including server cache, based on a heat map of data to lower latency and improve performance.

Figure 1-12 shows how the Hortonworks software stack is built on top of the IBM Spectrum Scale file system.

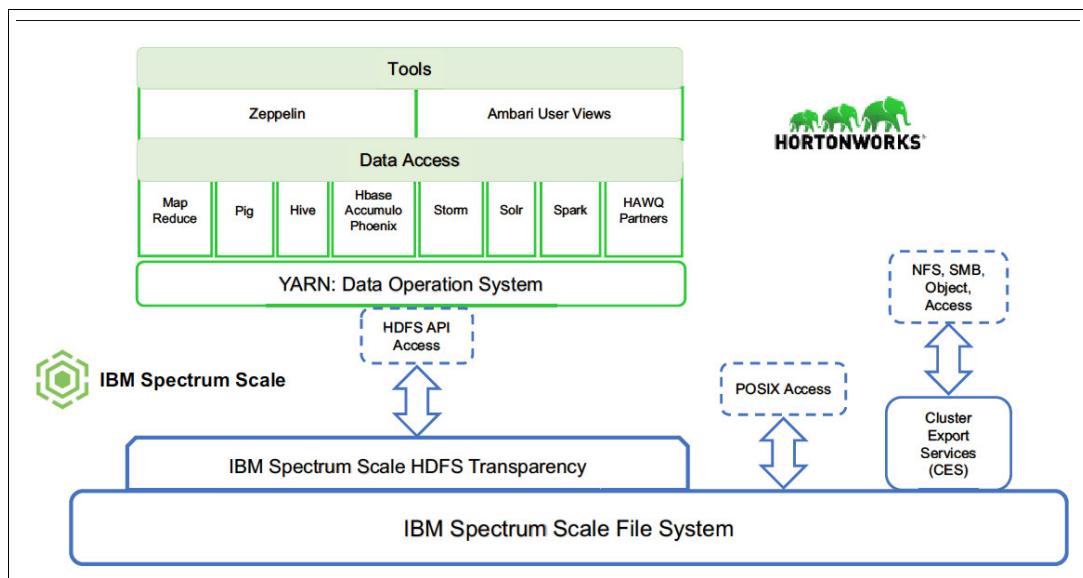


Figure 1-12 Hortonworks software stack on top of the IBM Spectrum file system

Employing HDP with IBM Spectrum Scale to optimize your data architecture can provide many cost benefits because the typical enterprise data warehouse dedicates 70% of its storage volume to unused data and 55% of its processing power to low-value extract, transform, and load (ETL) workloads.

Here are the top benefits of using IBM Spectrum Scale with HDP:

- ▶ Extreme scalability with a parallel file system architecture

IBM Spectrum Scale is a parallel architecture. With a parallel architecture, no single metadata node can become a bottleneck. Every node in the cluster can serve both data and metadata, enabling a single IBM Spectrum Scale file system to store billions of files. This architecture enables clients to grow their HDP environments seamlessly as the data grows. Additionally, one of the key value propositions of IBM Spectrum Scale, especially with IBM ESS, is running diverse and demanding workloads, plus the ability to create an active archive tier.

- ▶ A global namespace that can span multiple Hadoop clusters and geographical areas

Using IBM Spectrum Scale global namespace, clients can create active and remote data copies and enable real-time, global collaboration. This namespace enables global organizations to form data lakes across the globe and host their distributed data under one namespace.

IBM Spectrum Scale also enables multiple Hadoop clusters to access a single file system while still providing all the required data isolation semantics.

The IBM Spectrum Scale Transparent Cloud Tiering feature can archive data into a S3/SWIFT compatible cloud object storage system, such as IBM Cloud™ Object Storage or Amazon S3, by using the powerful IBM Spectrum Scale information lifecycle management (ILM) policies.

- ▶ A reduced data center footprint with the industry's best in-place analytics

IBM Spectrum Scale has the most comprehensive support for data access protocols. It supports data access by using NFS, SMB, Object, POSIX, and the HDFS API. This feature eliminates the need to maintain separate copies of the same data for traditional applications and for analytics.

- ▶ True software-defined storage (SDS) that is deployed as software or as a pre-integrated system

You can deploy IBM Spectrum Scale as software directly on commodity storage-rich servers running the HDP stack, or deploy it as part of a pre-integrated system by using the IBM ESS. Clients can use software-only options to start small while still using enterprise storage benefits. With IBM ESS, clients can control cluster sprawl and grow storage independently of the compute infrastructure. IBM ESS uses erasure coding to eliminate the need for the three-way replication for data protection that is required with other solutions.

Scalability with a parallel file system architecture

IBM Spectrum Scale delivers scalable capacity and performance to handle demanding data analytics, content repositories, and technical computing workloads.

Storage administrators can combine flash, disk, cloud, and tape storage into a unified system that is higher performing and lower cost than traditional approaches. With thousands of customers and more than 15 years of demanding production deployments, IBM Spectrum Scale is a file system that can adapt to both application performance and capacity needs across the enterprise. By including IBM Spectrum Scale in the software-defined infrastructure (SDI), organizations can streamline data workflows, help improve service, reduce costs, manage risk, and deliver business results while positioning the enterprise for future growth.

Figure 1-13 shows some connections that are provided by IBM Spectrum Scale.

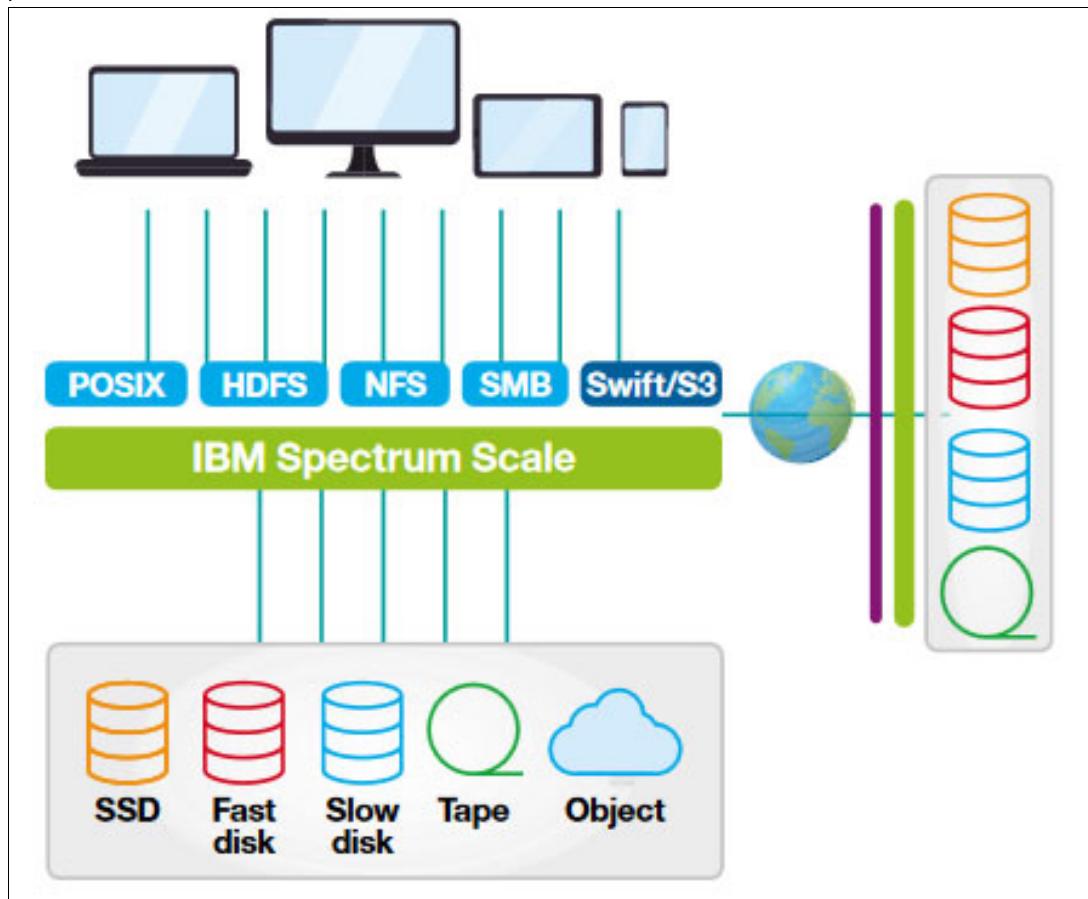


Figure 1-13 IBM Spectrum Scale

1.3.3 IBM Elastic Storage Server

IBM ESS is a modern implementation of SDS that combines IBM Spectrum Scale running on POWER8® processor-based I/O-intensive servers that are connected to many high-density, high-performance, and highly available dual-ported storage enclosures. This implementation eliminates data silos, simplifies storage management, and delivers high performance.

A key advantage for IBM ESS is its ability to lower capacity requirements. IBM ESS requires only 30% extra capacity to offer data protection benefits that are similar to HDFS triple replication. IBM Power Systems servers along with the IBM ESS offer the most optimized hardware stack for running analytics workloads. Clients can enjoy up to three times reduction of storage and compute infrastructure by moving to IBM ESS and IBM Power Systems servers compared to commodity scale-out x86 systems.

By consolidating storage requirements across your organization by using IBM ESS, you can reduce inefficiency, lower acquisition costs, and support demanding workloads.

IBM ESS with HDP allows data movement between different systems for data analysis. This movement is faster and more lightweight, which eliminates the need to copy data from a POSIX-based file system to the HDFS because of the unified namespace that is offered by IBM Spectrum Scale.

IBM ESS hardware and software solution benefits

In an enterprise big data solution environment, Hadoop generates and consumes data from the data lake. Moving data to and from the HDFS (by running `distcp`) is arduous, time-consuming, and consumes unnecessary disk space.

IBM ESS includes a software RAID function that eliminates the need for the classic replication of three factors for data availability, protection, and fast access to the data by the Hadoop HDFS. Instead, IBM ESS requires just an extra of 30% of capacity to offer similar or better data protection.

The IBM ESS solution supports a massive volume of data, which is always a requirement for a real-world data lake. A single ESS can deliver up to 40 Gbps of throughput and multiple ESS's can scale out to exabytes of storage to support a massive expansion of the business. This growth of data can be managed and tiered across practically any kind of storage, disks technologies, cloud, and tape.

The core of the IBM ESS is IBM Spectrum Scale, which manages all disks, data, and file system-related tasks on the IBM ESS solution in a fully POSIX-supported mode. The file systems can be accessed by any UNIX based and Linux based operating system.

IBM Spectrum Scale also supports the SMB protocol, which enables the growth of various servers that can access the data inside the namespace (file system) without needing a specific client program.

The hardware part of the IBM ESS solution is composed of IBM Storage enclosures with high-density solid-state drives (SSDs) or hard disk drives (HDDs) and redundant communication adapters and electric connectors. Data management and data transport and distribution to the initiators servers is done by POWER8 processor-based servers. This solution is a highly available and high-performance solution.

Figure 1-14 shows the table of IBM ESS hardware and software stack configurations.

IBM Elastic Storage Server		
Elastic Storage Server Models		
GS1S, GS2S, GS4S	Storage units	5147-024: 24-slot 2U enclosure for 2.5" SSDs
	Number of drive enclosures	1, 2 or 4
	Drive capacity	3.84 TB or 15 TB 2.5" SSDs
	Number of drives	24 - 96 SSDs
	Maximum usable capacity per ESS	60 TB (GS1S) to 1.1 PB (GS4S)
GL1S, GL2S, GL4S, GL6S	Storage units	5147-084: 84-slot 5U enclosure for 3.5" HDDs
	Number of drive enclosures	1, 2, 4 or 6
	Drive capacity	4, 8 or 10 TB NL-SAS 3.5" HDDs
	Number of drives	82 - 502 HDDs
	Maximum usable capacity per ESS	600 TB (GL1S) to 3.9 PB (GL6S)
GH12, GH14, GH24	Storage units	5147-024: 24-slot 2U enclosure for 2.5" SSDs & 5147-084: 84-slot 5U enclosures for 3.5" HDDs
	Number of drive enclosures	Qty=1 or 2, 24 drive enclosures for SSDs Qty=2 or 4, 84-slot drive enclosures for HDDs
	Drive capacity	3.84 TB or 15 TB 2.5" SSDs 4, 8 or 10 TB NL-SAS 3.5" HDDs
	Number of drives	24 - 48 SSDs & 168 or 336 HDDs
	Maximum usable capacity per ESS	60 TB SSD + 1 PB HDD (GH14) to 530 TB SSD + 2.5 PB HDD (GH24)
GL1C, GL2C, GL4C, GL6C	Storage units	5147-106: 106-slot 4U drive enclosure for 3.5" HDDs
	Number of drive enclosures	1, 2, 4 or 6
	Drive capacity	10 TB NL-SAS HDDs
	Number of drives	104 - 634 HDDs
	Maximum usable capacity per ESS	0.7 PB (GL1C) to 4.6 PB (GL6C)
Capacities shown are approximate using 8+2P erasure coding and a single Data + Metadata pool. Example configurations are shown for minimum and maximum capacity figures- other intermediate options are also available.		
System information		
Systems	IBM Power 5148-22L ESS Data Server (two per Building Block) IBM Power 5148-21L ESS Management Server (one per cluster)	
Interconnects	Ethernet: 10GbE, 40GbE, 100GbE Infiniband: 56 Gbps FDR, 100 Gbps EDR	
Operating system	Red Hat Enterprise Linux (RHEL) Little Endian	
Software	IBM Spectrum Scale: Data Access Edition or Data Management Edition xCAT	
Supported RAID levels	Erasure coding: 3-way or 4-way mirroring, 8+2P or 8+3P, or a combination	

Figure 1-14 IBM ESS hardware and software stack configuration

1.3.4 IBM Spectrum Scale HDFS Transparency Connector

IBM Spectrum Scale HDFS Transparency Connector enables applications to use a standard HDFS client to access IBM Spectrum Scale through native remote procedure call (RPC) requests. All data transmission and metadata operations in HDFS use the RPC mechanism and are processed by NameNode and DataNode services within HDFS. IBM Spectrum Scale HDFS Transparency Connector integrates both the NameNode and the DataNode services, and responds to the requests from HDFS clients. HDFS clients can continue to access IBM Spectrum Scale seamlessly, just as it does with HDFS.

Figure 1-15 shows the overall dataflow between IBM Spectrum Scale, storage, and the compute nodes.

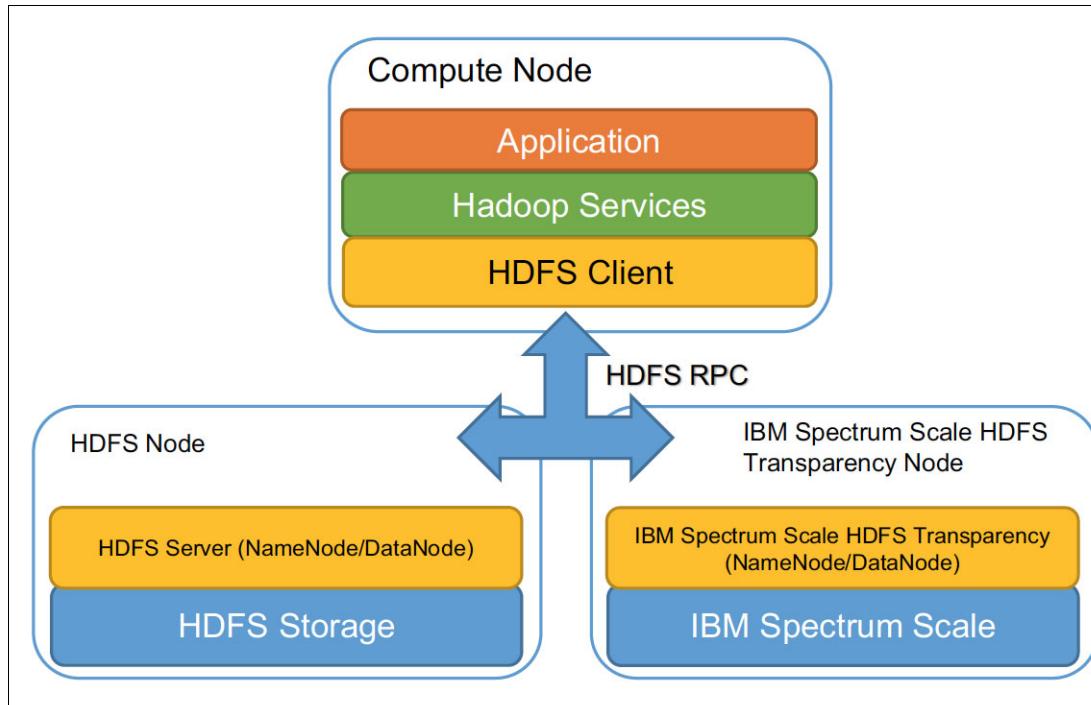


Figure 1-15 Overall dataflow between IBM Spectrum Scale, storage, and the compute nodes

Here are the key advantages of IBM Spectrum Scale HDFS Transparency Connector:

- ▶ Reduces the data center footprint.
- ▶ IBM Spectrum Scale provides in-place analytics for your Hadoop data, which eliminate the need for several copies of the same data or large migrations of data between HDFS and the POSIX file system.
- ▶ Control cluster sprawl: IBM Spectrum Scale delivers federation across both IBM Spectrum Scale and HDFS clusters, which turns isolated data lakes into a data ocean while still maintaining needed separations.
- ▶ Makes HDFS access enterprise-ready.
- ▶ IBM ESS (on a preintegrated IBM Spectrum Scale based system) adds storage functions that are necessary to the enterprise (for example, encryption, disaster recovery (DR), software, and RAID) to your Hadoop setup.
- ▶ An IBM Spectrum Scale Client is not needed on every Hadoop node. The HDFS client can access data on IBM Spectrum Scale as it does with HDFS storage.
- ▶ Improved security management by Kerberos authentication and encryption for RPCs.
- ▶ Support for more Hadoop components or HDFS-compliant APIs or commands (for example, `discp` and `webhdfs`).
- ▶ IBM Spectrum Scale provides an HDFS connector and allows existing Hadoop applications to run directly on IBM Spectrum Scale.

IBM Spectrum Scale and HDFS Transparency differences

Here are other key HDFS Transparency and IBM Spectrum Scale differences to note:

- ▶ If one file is set with an access control list (ACL) (POSIX or NFSv4 ACLs), IBM Spectrum Scale HDFS Transparency Connector does not provide the interface to disable the ACL check at the IBM Spectrum Scale HDFS Transparency Connector layer. If you want to disable the ACL for one file, the only way is to remove the ACL.
- ▶ IBM Spectrum Scale provides its own caching mechanism that does not support HDFS caching. Caching that is done by IBM Spectrum Scale is more optimized and controlled, especially when you run multiple workloads. The interface `hdfs cacheadmin` is not supported by IBM Spectrum Scale HDFS Transparency Connector.
- ▶ The NFS Gateway from native HDFS is not needed by IBM Spectrum Scale HDFS Transparency Connector. If your applications require an NFS interface, use the IBM Spectrum Scale protocol Cluster Export Services (CES) and NFS for better performance, scaling, and automatic NFS server failover.
- ▶ IBM Spectrum Scale provides multiple protocol interfaces, including POSIX, NFS, and SMB. Customers can use IBM Spectrum Scale Protocol for NFS to access the data.
- ▶ The option `distcp -diff` is not supported for snapshot over IBM Spectrum Scale HDFS Transparency. Other options from `distcp` are supported.
- ▶ The interface from `hdfs dfs` is supported, but others (such as `hdfs fsck`) are not needed for IBM Spectrum Scale HDFS Transparency Connector.

1.4 When to use a GPU

Traditionally, most computing tasks were performed on the CPU. However, there is a certain class of tasks that best benefits from using GPUs. CPUs are built for general-purpose tasks and work on versatile computing tasks. Particularly, single-threaded workloads work well on a CPU. However, there are computing tasks that are parallel in nature. Tasks such as large matrix multiplication can be split into many parallel jobs, and acting on different data elements is more efficient in a GPU. GPUs typically contain many cores with lower frequency compared to CPUs, which have fewer cores (typically less than 80 processing threads per socket) with higher frequency. GPUs are efficient in handling floating point operations.

DL problems are massively parallel operations that perform many matrix operations and operate on a massive amount of data. The efficiency of the deep neural networks is based on the large data availability and the depth and width of the neural architecture. A typical training phase of a deep neural network consists of forward propagation, gradient computation, and backward propagation, which constitutes one iteration of training. You must perform several thousands of these iterations to create an efficient DL model. Running such massively parallel tasks for many iterations on the CPU takes several days. Data scientists and DL researchers must perform several experiments to arrive at a good neural model or neural architecture. The DL tasks are floating-point rich and massively parallel, which makes them a good candidate for the GPU.

However, using a GPU does not mean that the CPU is not used in these computations. The GPU always works with the CPU. The CPU works on the parts of the computation that are single-threaded or serial in nature, and off loads the bulk of the parallel tasks to the GPU. The GPU on completion of these parallel jobs returns the control flow back to the CPU. Choosing a GPU or CPU for a computation task depends on the nature of the workload that is run.

A typical DL training task is broadly divided into three phases.

- ▶ Data loading
- ▶ Data transformation
- ▶ Training

Generally, the data loading and data transformation tasks are completed at the CPU because most of the libraries that handle this task are single-threaded and optimized for the CPU. The training is carried out in the GPU, where the program spends more than 90% of the total application run time. However, recently there have been several advancements in the libraries for data loading and transformation, which make those tasks suitable for the GPU environment. When we reach this point, we have almost 100% of the application running on the GPU and benefitting from the parallelism it offers.

Most of the servers that are available in the accelerator market have one to many GPUs. These GPUs are either attached to the server through a PCIe interface or a NVLink interface, or are on an external drawer that is connected to the CPU. The nature of the workload can help determine the number of GPUs that you need for the computation. DL problems are data parallel in nature, where different segments of data can be operated in parallel across different GPUs. Considering that DL problems need enormously large amounts of data, it is imperative that you employ multiple GPUs to work on the problem in parallel, thus expediting the speed of a training task.

IBM Power Systems servers support the most powerful GPUs on its platform. The NVIDIA Tesla V100 Tensor Core is the most advanced data center GPU ever built to accelerate AI, high-performance computing (HPC), and graphics. It is powered by the NVIDIA Volta architecture, comes in 16 GB and 32 GB memory configurations, and offers a performance of up to 100 CPUs in a single GPU. Data scientists, researchers, and engineers can now spend less time optimizing memory usage and more time designing the next AI breakthrough. With 640 Tensor Cores, Tesla V100 is the world's first GPU to break the 100 teraflops (TFLOPS) barrier of DL performance. This new next generation of NVIDIA NVLink GPUs connects multiple V100 GPUs at up to 300 GBps to create the world's most powerful computing servers. AI models that consumed weeks of computing resources on previous systems can now be trained in a few days. With this dramatic reduction in training time, a whole new world of problems is now solvable with AI.

GPUs are fast compared to CPUs. So, it is essential to make sure that there are no bottlenecks in the path where the computations are sent to the GPU. If there are bottlenecks, the GPUs *starve* or are underused. DL training takes a batch of data from the local memory of the server or from a data lake that is attached to the server and loads the data to the main memory of the server CPU. From there, it pushes the data to the GPU through the available communication medium. An AI solution architecture should have an efficient data lake, a superior CPU memory, and a high-bandwidth network channel from the CPU to the GPU to avoid any bottlenecks for the GPU computation to occur efficiently.

The use of NVIDIA NVLink in IBM Power Systems servers helps to achieve a bandwidth of around 300 GBps between CPU and ALL GPUs, unlike PCIe-attached GPUs, which can reach a bandwidth of only 64 GBps. This is one of the major advantages of performing AI tasks on Power Systems servers. For example, handling larger DL models that do not fit inside the GPU memory is managed across both the CPU and GPU memory. In such scenarios, the artifacts of the training must be constantly moved between the CPU and the GPU memory at high speed. These are the AI applications of the future because providing more depth and width to a neural model effectively produces better accuracy. However, a model with more depth and width occupies more memory. The higher bandwidth NVLink helps heavily in this scenario.

Figure 1-16 provides the major specifications of the Tesla V100.

Tesla V100 SXM2	
GPU Architecture	NVIDIA Volta
NVIDIA Tensor Cores	640
NVIDIA CUDA Cores	5120
Double-Precision Performance	7.8 TFLOPS
Single-Precision Performance	15.7 TFLOPS
Tensor Performance	125 TFLOPS
GPU Memory	32GB/16GB HBM2
Memory Bandwidth	900GB/sec
ECC	Yes
InterConnect Bandwidth	300GB/sec
System Interface	NVIDIA NVLink
Form Factor	SXM2
Max Power Consumption	300 W
Thermal Solution	Passive
Compute APIs	CUDA, DirectCompute, OpenCL, OpenACC

Figure 1-16 NVIDIA Volta GPU specifications

1.5 Hortonworks Data Platform GPU support

HDP 3.0 with Hadoop 3.1 provide new capabilities for enterprises, which include faster application development through containerization, erasure coding, namenode federation, and better support for DL applications by managing GPU resources as a first class asset. In this section, the focus is the introduction of GPU support as a native asset and its benefits.

GPUs have many cores and can handle massive but simple tasks simultaneously. A CPU has fewer cores, but are powerful and can complete complex tasks. For use cases such as matching images, real-time vision processing, and aerial navigation, GPUs are the best option. DL work runs much faster on GPUs because of the higher core density.

The main benefits of using GPUs within the HDP cluster nodes are:

- ▶ No need to set up separate clusters.
- ▶ GPU discovery and configuration: The admin can configure GPU resources and architectures on each node.
- ▶ GPU isolation and monitoring: After you start a task with GPU resources, NodeManager should isolate and monitor the task's resource usage.
- ▶ GPU scheduling: The YARN scheduler should count a GPU as a resource type just like CPU and memory.

1.5.1 Native GPU support

HDP 3.1 GPU support is available on YARN as a native asset so that administrators and operators may schedule and operate with GPU resources. Before there was native GPU support, users used node-labels to partition clusters to use GPUs, which put machines that were equipped with GPUs in a different partition. The users had to submit jobs that required GPUs to the partition.

Many applications need expensive and specific hardware, such as GPUs, on which to run. So, applications that had to run on a specific node in the cluster could not do so in YARN. It was only possible by using node labeling, which has its own specific requirements. Node labeling works only for a decentralized or centralized per-node configuration, but not for both together. The only endpoint for a centralized node should be a command-line interface (CLI) or REST. If you wanted to change labels, it was complex and difficult. Without native and more comprehensive GPU support, there is no isolation of GPU resources. Sometimes, even multiple tasks compete for a GPU resource simultaneously, which can cause task failures and GPU memory exhaustion. That is why the need for native GPU support was so important.

1.5.2 GPU discovery

To discover how many GPUs you have on your system, run **nvidia-smi**. Figure 1-17 shows the output of the command.

```
[root@p460a01 ~]# nvidia-smi
Fri Dec 14 19:34:46 2018
+-----+
| NVIDIA-SMI 410.72      Drive
| Version: 410.72        CUDA Version: 10.0      |
+-----+
| GPU  Name      Persistence-M | Bus-Id     Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|-----+-----+-----+-----+-----+-----+
| 0  Tesla V100-SXM2... On   | 00000004:04:00.0 Off |          0 |
| N/A  32C    PO    42W / 300W | 10MiB / 31744MiB | 0%       Default |
+-----+-----+-----+-----+-----+-----+
| 1  Tesla V100-SXM2... On   | 00000004:05:00.0 Off |          0 |
| N/A  36C    PO    42W / 300W | 10MiB / 31744MiB | 0%       Default |
+-----+-----+-----+-----+-----+-----+
| 2  Tesla V100-SXM2... On   | 00000035:03:00.0 Off |          0 |
| N/A  31C    PO    40W / 300W | 10MiB / 31744MiB | 0%       Default |
+-----+-----+-----+-----+-----+-----+
| 3  Tesla V100-SXM2... On   | 00000035:04:00.0 Off |          0 |
| N/A  37C    PO    42W / 300W | 10MiB / 31744MiB | 0%       Default |
+-----+-----+-----+-----+-----+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name           Usage      |
|-----+-----+-----+-----|
| No running processes found            |
+-----+
```

Figure 1-17 The nvidia-smi command output

You might want to discover your GPUs automatically. This task is possible with HDP 3.0 and Hadoop 3.1.

1.5.3 GPU isolation and monitoring

When multiple applications are using a single GPU's resources on the same machine, you might need to use *GPU isolation*. If multiple processes use a single GPU, a process must wait until the current one finishes before it can start, which can cause delays. Another issue might be frameworks like TensorFlow that aggressively try to use the GPU's resources. If you assign two TensorFlow applications to a single GPU, both of them cannot run because the GPU runs out of memory.

This is why you need GPU isolation. The application performance should not be affected by a lack of GPU resources. The granularity for GPUs is per-GPU device, which is achieved by using control groups (cgroups) or Docker to enforce isolation.

1.5.4 GPU scheduling

With GPU scheduling as a native YARN resource, it is much easier for administrators to allocate the needed resources to applications that request them. GPU scheduling ensures that specific applications always have sufficient GPU resources and that applications may be denied if there are resource shortages. It is common that a model training process exceeds the maximum available GPU memory (OutOfMemory errors). In some specific use cases, the required amount of memory can be large, which can be a serious problem if you do not have your training session isolated from the rest of your infrastructure.

Consider an example without a resource manager. If you have a model running in production mode and start a poorly configured training session that requests all the available GPU memory, at least one of the two applications will fail because of the memory shortage. YARN is useful in an ML environment because its scheduler and resource management capabilities are essential when applied to GPUs.

1.5.5 Hortonworks Data Platform 3 and YARN container with GPU support

YARN can support many types of use cases. Features like GPU pooling and isolation are expanding DL frameworks like Caffe, Pytorch, and TensorFlow so that GPUs can be shared as a resource among data scientists. The YARN native support for managing GPU isolation enables running complex DL data analytics in the same Hadoop cluster where ETL, batch jobs, TensorFlow ML loads, and diversified loads are running in a single cluster that is powered by YARN. You do not have to set up separate partitions, machines, or clusters for GPU workloads. Native GPU support in HDP 3 facilitates the usage of Docker or cgroup containers where one or more GPU cores are assigned and isolated for those containers. NVIDIA plug-in support is integrated into YARN to support containers so that GPU cores can be used exclusively for those containers.

1.6 Linux on Power

IBM Power Systems servers offer the best infrastructure solution for deploying IBM Watson Studio Local. The IBM Power System LC922 and IBM Power System LC921 servers provide the ideal base for the control, storage, and compute (non-GPU) components of IBM Watson Studio Local, and the IBM Power System AC922 server offers an optimized GPU-attached environment for the compute components of IBM Watson Studio Local. The Power LC922 server's superior Spark performance and storage capacities paired with the Power AC922 server's leadership performance for GPU-accelerated algorithms deliver the perfect mix of performance, scalability, and price-point for a IBM Watson Studio Local deployment.

Built specifically for enterprise AI, the POWER9 processor is the only processor with state-of-the-art I/O subsystem technology, including next-generation NVIDIA NVLink, PCIe Gen4, and IBM OpenCAPI™. These interfaces give the POWER9 processor a memory bandwidth superhighway that can run ML and DL applications with unmatched performance and scalability, handling even very large data sets with ease.

IBM estimates that Watson can process up to 60 million pages of plain text per second. Plain text is unstructured, much like about 80% of all existing information that current IT systems must process. The surprising part is that Watson can make sense of almost any free-form information source whether it is someone speaking or a dictation of someone's handwritten notes. It can learn and quickly. It is positioned to assist medical diagnosis for patients, revolutionize customer service, create recipes for chefs, and a number of other interesting and previously insurmountable challenges. Some of these challenges are critically important and others are simply fun and challenging applications that might well become highly strategic in the future.

A Power Systems server's enhanced security features and lack of security vulnerabilities allows IT managers to avoid costly consequences of security breaches, such as:

- ▶ Strengthening existing IT security and carrying out more training
- ▶ Contacting those whose records might have been exposed
- ▶ Legal action that is taken by people who might have suffered a financial loss
- ▶ Impacts on share price
- ▶ Costs to regain market position

Linux on Power is the same Linux as on x86 servers. It does not cost more to run Linux on Power. In fact, in most cases the total cost of acquisition and the total cost of ownership (TCO) favor running it on Power Systems servers.

1.7 Client use cases

The following sections describe several client use cases.

1.7.1 Asian job-hunting services company

Resources: HDP, IBM Spectrum Scale storage, IBM PowerAI, and Power Systems servers

An Asia-based job hunting services company that offers services for personal staffing, recruitment, and new graduate support was looking for a new cloud-based solution to help support their daily business activities. Due to the necessity of storing and analyzing company information, recruiting activities, job postings, and hiring history, the company was adding more than a terabyte of data per day to their data volume, which was unsustainable growth with their current solution. In addition to upgrading their data volume capabilities, they also needed to use AI to manage more complex data, including handling audio, video, and still images.

For a job-hunting services company, speed and accuracy of matching data is critical. They decided to implement HDP, IBM Spectrum Scale storage, IBM PowerAI, and Power Systems servers to support their Hadoop and Spark workloads because of their fast performance, reliability, and scalability. The company deployed IBM Spectrum Scale storage with the File Placement Optimizer so that they could start small by adding the software on Hadoop nodes while still achieving faster data import and in-place analytics than if using HDFSs. If needed, they also can add IBM ESS based shared storage in the future within the same Hadoop cluster. With IBM ESS, they can expect to use up to 60% less storage.

The solution provides them with 70% faster processing than their previous solution while adding increased analytical ability with a smaller storage footprint.

1.7.2 Large bank

Resources: HDP, IBM ESS, Power Systems servers, and IBM Spectrum Scale

A large, award-winning bank with over 16,000 employees had the vision of providing a data-driven customer experience to their millions of customers. Their goal was to deliver near real-time data visualizations, new services, and personalized offers to their customers one to one. They also needed to develop an open platform to import all structured and unstructured data from a multitude of different sources and store and extract analytical insights from the data.

Their current solution could not use data science to identify patterns within data sets and prototype new data models in a timely manner. To augment their current analytical capabilities, they chose HDP on Power Systems servers with IBM ESS. They ran a HiBench workload by using IBM ESS disk utilization to the maximum, and it performed significantly better than the Intel-based solution. With Hortonworks running on Power Systems servers and the implementation of IBM ESS and IBM Spectrum Scale, they optimized the management of their data lake while using less than half the number of computer nodes at a lower cost.

1.7.3 South American IT services provider

Resources: HDP

A leading South American IT services provider that specializes in business processes, help desk, and IT outsourcing services, needed to improve their cloud infrastructure to offer database services to their largest hosting customer. They also had the vision of instituting a new private cloud for analytical and AI-based services for their clients.

They decided on HDP after seeing proof of concepts (PoCs) for analytics, AI, and open source database cloud services and the cost-effectiveness compared to other solutions. The Power platform fit their needs by providing an open source architecture with 1.7x the performance advantage of other solutions and high-performance storage capacity.

1.7.4 Governmental agency

Resources: Power Systems servers, HDP, IBM Streams, and IBM Cognos® Analytics

A Middle Eastern governmental organization with over a million employees needed a data-driven decision-making platform to perform advanced analytics and help effectively manage crises.

They decided on the Power platform because it was an end to end solution that can better capture the necessary data and analyze key trends to better predict crises and prepare a resolution.

1.7.5 European IT services company

Resources: HDP, POWER9 processor-based servers, IBM Watson Studio Local, and IBM ESS

A prominent European IT services company servicing over 120 million accounts needed a way to improve their customer experience by using data-driven decision-making. They decided on a data lake strategy, but needed a new infrastructure to recognize quickly data

patterns and leverage data science to create automatically models. They required an open source, cloud-based platform that improved performance, reduced cost, and could handle all structured and unstructured data from different sources.

The company decided on HDP on POWER9 processor-based servers because of its cost reduction and scalability. They also added IBM Watson Studio Local to their HDP platform to take advantage of all of its data science tools and enable their developers to create AI-based solutions.



Integration overview

This chapter describes how to integrate the various products to build a scalable solution.

The following topic are described in this chapter:

- ▶ Architecture overview
- ▶ System configurations
- ▶ Deployment options
- ▶ IBM Watson Machine Learning Accelerator and Hortonworks Data Platform
- ▶ IBM Watson Studio Local with Hortonworks Data Platform
- ▶ IBM Watson Studio Local with IBM Watson Machine Learning Accelerator
- ▶ IBM Spectrum Scale and Hadoop Integration
- ▶ Security

2.1 Architecture overview

This architecture overview describes the software components that are used to build the artificial intelligence (AI) environment as the platform, which includes the AI frameworks; the software that is used to build a multi-tenant data science environment; the distributed computing, training, and inference environment; and a tiered data management environment (Figure 2-1). This architecture addresses some of the bigger challenges facing developers and data scientists by cutting down the time that is required for AI system training and simplifying the development experience.

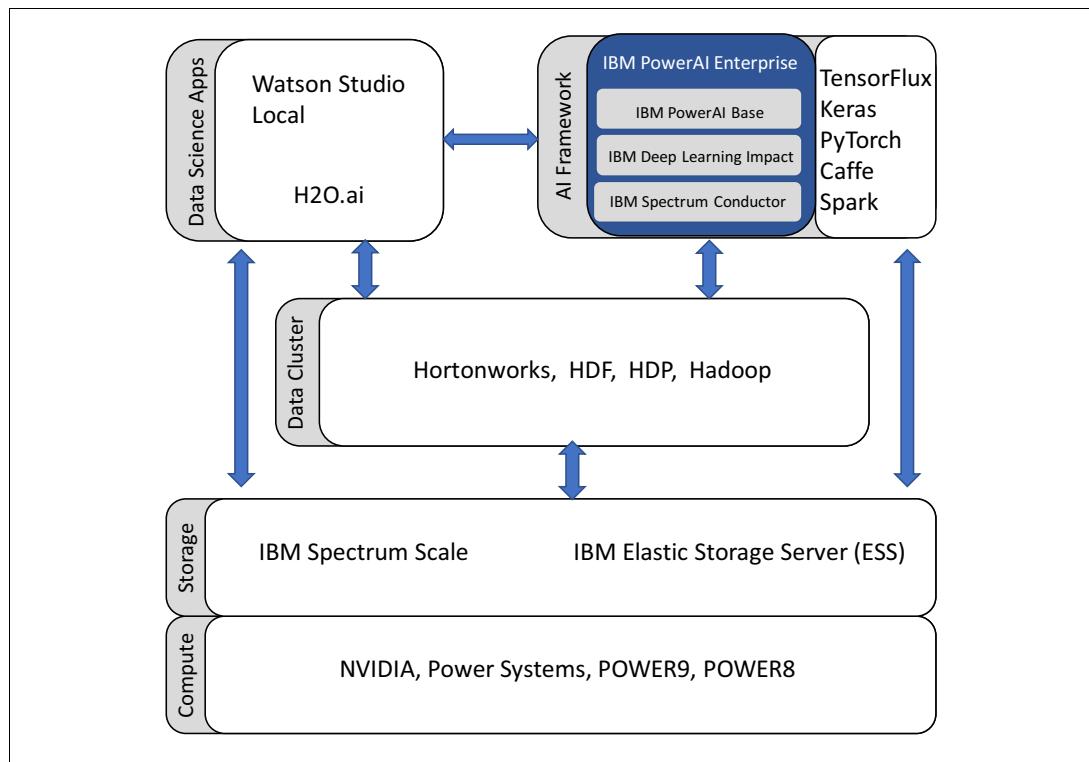


Figure 2-1 How the components fit together

During the lab testing for this publication, the products and versions that were used are listed in Table 2-1.

Table 2-1 Software levels

Product	Version
IBM PowerAI	1.5.4
IBM Watson Machine Learning Accelerator	1.1.2
H2O Driverless AI	1.4.2
IBM Watson Studio Local	1.2.1.0
Hortonworks Data Platform (HDP)	2.6.5.0
Hortonworks DataFlow (HDF)	3.1.2.0
IBM Elastic Storage Server (IBM ESS)	5.3.1.1

Product	Version
IBM Spectrum Scale	5.0.1.2
IBM Power Systems servers	POWER9 and POWER8 processors

2.1.1 Infrastructure stack

The infrastructure includes compute nodes, GPU accelerators, a storage solution, and networking.

Compute nodes

The resource of primary interest for AI is the GPUs, and they are on the IBM Power Systems server nodes:

- ▶ IBM Power System AC922 servers feature POWER9 CPUs and support 2 - 6 NVIDIA Tesla V100 GPUs with NVLink providing CPU:GPU bandwidth speeds of up to 300 GBps. This system supports up to 2 TB total memory. For more information, see [IBM Power System AC922](#).
- ▶ IBM Power System S822LC for High-Performance Computing (Power S822LC for HPC) servers feature POWER8 CPUs and support 2 - 4 NVIDIA Tesla P100 GPUs with NVLink GPUs providing CPU:GPU bandwidth speeds of 64 GBps. For more information, see [IBM Power System S822LC for High Performance Computing](#).

Table 2-2 provides the compute configuration system details that are provisioned on the lab environment for this book.

Table 2-2 System details

Server type	GPU	OS	CPU	Memory	Drive
Power AC922 server	Four NVIDIA V100 GPUs	RHEL 7.5	40 POWER9 cores @3.8 GHz	1 TB	Two 1.92 TB solid-state drives (SSDs)
Two Power LC922 servers	N/A	RHEL 7.5	40 POWER9 cores @3.8 GHz	512 GB	Two 128 GB SATA DOM drives Four 8 TB hard disk drives (HDDs)
Two Power S822LC for Big Data servers	N/A	RHEL 7.5	20 POWER8 cores @3.5 GHz	512 GB	Two 128 GB SATA DOM drives Four 1.92 TB SSDs Two 1.6 TB NVMe SSDs
Power S822LC for HPC server	N/A	RHEL 7.5	20 POWER8 cores @3.5 GHz	1 TB	Two 3.8 TB SSDs One 2.9 TB NVMe SSD

GPU

There are up to four NVIDIA Tesla V100 GPUs integrated into an aircooled Power AC922 server and up to six V100 per water cooled AC922. This model is the most advanced data center GPU ever built to accelerate AI, high-performance computing (HPC), and graphics. It is based on the NVIDIA Volta architecture, comes in 16 GB and 32 GB configurations, and offers a performance of up to 100 CPUs in a single GPU.

Storage

To support the variety and velocity of data that is used and produced by AI, the storage system must be intelligent, have enough capacity, and be highly performant. Key attributes include tiering and public cloud access, multiprotocol support, security, and extensible metadata to facilitate data classification. Performance is multi-dimensional for data acquisition, preparation, and manipulation; high-throughput model training on GPUs; and latency sensitive inference. The type of storage that is used depends on the location of the data and the stage of processing for AI. Corporate and older data usually is in an organizational data lake in Hadoop Distributed File System (HDFS).

IBM ESS combines IBM Spectrum Scale software with IBM POWER8 processor-based I/O-intensive servers and dual-ported storage enclosures. IBM Spectrum Scale is the parallel file system at the heart of IBM ESS and scales system throughput as it grows while still providing a single namespace, which eliminates data silos, simplifies storage management, and delivers high performance.

For more information, see [IBM Elastic Storage Server](#).

The storage configuration that is allocated for the integration that is used in this publication is shown in Table 2-3.

Table 2-3 IBM ESS GS2

Specification	Details
Machine type and model number	5126-GS2
Serial number	218E54G
Drive type	Forty-eight 400 GB SSDs
File system capacity	11.17 TB
File system block size	16 MB

Networking

The exact architecture, vendors, and components that are needed to build the network subsystem depend upon the organization's preference and skills. InfiniBand or high-speed Ethernet and a network topology that allows both north and south (server to storage) traffic and can also support east and west (server to server) traffic are required. Adopting a topology that extends to an InfiniBand Island structure enables the training environment to scale for large clusters. An adequate network subsystem with necessary throughput and bandwidth to connect the different tiers of storage is required.

IBM ESS offers network adapter options. Three PCI slots are reserved for SAS adapters and one PCI slot is configured by default with a 4-port 10/100/1000 Ethernet adapter for management. Three other PCIe3 slots are available to configure with any combination of Dual-Port 10 GbE, Dual-Port 40 GbE, or Dual-Port InfiniBand PCI adapters.

The following networking interfaces are allocated for the integration that is deployed in this publication:

- ▶ 10 Gb administration network
- ▶ 56 Gb EDR InfiniBand network

2.2 System configurations

A small starter configuration for IBM Watson Studio Local, IBM Watson Machine Learning Accelerator and HDP depend on the needs of the organization. For example, choosing whether to use a 3-node or 9-node IBM Watson Studio Local cluster depends on the number of users that need access to the system. If an organization expects 10 - 30 users to use the system concurrently, then a 3-node configuration is sufficient. However, if more than 30 concurrent users are expected, then a 9-node cluster should be put in place.

IBM Watson Machine Learning Accelerator is the cluster manager, and can set up service-level agreements (SLAs) between the instances and usage of resources. IBM Watson Machine Learning Accelerator provides many optimizations that accelerate performance; improve resource utilization; and reduce installation, configuration, and management complexities. HDP integration with IBM Watson Studio Local and IBM Watson Machine Learning Accelerator presents a unique mix of technologies that improve the work of a data scientist. This integration makes your work with big data more efficient, and it makes data science more accessible and scalable by bringing all enterprise data to a new level of accurate prediction.

2.2.1 IBM Watson Machine Learning Accelerator configuration

IBM Watson Machine Learning Accelerator is optimized for the Power AC922 and the Power S822LC for High-Performance Computing servers. IBM Watson Machine Learning Accelerator is optimized to use IBM Power Systems servers with NVLink between the Power CPUs and NVIDIA GPUs which is not available on any other platform. IBM Watson Machine Learning Accelerator is supported on Power AC922 servers with NVIDIA Tesla V100 GPUs, and Power S822LC servers with NVIDIA Tesla P100 GPUs.

Powered by the revolutionary NVIDIA Volta architecture, the Tesla V100 GPU satisfies the most stringent demands of today's next-generation AI, analytics, deep learning (DL), and HPC workloads. The Tesla V100 GPU accomplishes these tasks by delivering 112 teraflops (TFLOPS) of deep-learning performance to accelerate compute-intensive workloads. It delivers over 40% performance improvements for HPC and over 4X deep-learning performance improvements over the previous-generation NVIDIA Pascal architecture-based Tesla P100 GPU.

The base system configuration is as follows:

- ▶ Two IBM POWER9 or POWER8 CPUs.
- ▶ 128 GB or more of memory.
- ▶ NVIDIA Tesla P100 or V100 with NVLink GPUs are required.
- ▶ NVIDIA NVLink interface to Tesla GPUs preferred.

Here are the software requirements:

- ▶ Red Hat Enterprise Linux 7.5 for POWER9, any subsequent updates
- ▶ Third-party software from NVIDIA, such as CUDA, CUDA Deep Neural Network (cuDNN), and NCCL for CUDA.

Table 2-4 and Table 2-5 list the minimum hardware and software system requirements for running IBM Watson Machine Learning Accelerator in a production environment. You might have extra requirements (such as extra CPU and RAM) depending on the Spark Instance Groups (SIGs) that run on the hosts, especially for compute hosts that run on workloads.

Table 2-4 Hardware requirements

Requirements	Management hosts	Compute hosts	Notes
RAM	64 GB	32 GB	In general, the more memory that your hosts have, the better performance is.
Disk space that is required to extract the installation files from the IBM Watson Machine Learning Accelerator installation package	16 GB (First Management host only)	N/A	None.
Disk space that is required to install IBM Spectrum Conductor	12 GB	12 GB	None.
Disk space that is required to install IBM Spectrum Deep learning impact	11 GB	11 GB	None.
Extra disk space (for Spark Instance Groups (SIG) package, logs, and other items.)	Might be 30 GB for a larger cluster	1 GB*N slots + sum of service package sizes (including dependencies)	The disk space requirements depend on the number of SIGs and the Spark application that you run. Long-running applications like notebooks and streaming applications can generate huge amounts of data that is stored in Elasticsearch.

Table 2-5 Software requirements

Hardware	Operating system	GPU software
POWER8 processor	RHEL 7.5 (ppc64le)	<ul style="list-style-type: none"> ▶ cuDNN 7.3.1library ▶ NVIDIA CUDA 10.0.130 ▶ NVIDIA GPU driver 410.72 ▶ Anaconda 5.2 ▶ NVIDIA NCCL 2.3.5
POWER9 processor with the security fix RHSA-2018:1374 - Security Advisory	RHEL 7.5 (ppc64le)	<ul style="list-style-type: none"> ▶ cuDNN 7.3.1 library ▶ NVIDIA CUDA 10.0 ▶ NVIDIA GPU driver 410.72 ▶ Anaconda 5.2 ▶ NVIDIA NCCL 2.3.5

To get the preferred performance from integrating Hadoop with IBM Watson Studio Local and IBM Watson Machine Learning Accelerator, start with choosing the correct hardware and software stack. The planning stage is vital in terms of determining the performance and the total cost of ownership (TCO) that is associated with it.

2.2.2 IBM Watson Studio Local configurations

This section describes what small starter configurations look like for IBM Watson Studio Local (formerly known as IBM Data Science Experience (IBM DSX)) deployed with an HDP datalake. It also describes advanced configurations.

IBM and Hortonworks worked together to integrate IBM Watson Studio Local with HDP.

The system requirements for IBM Watson Studio Local describe in detail the hardware and software requirements for 3-node, 7-node, and 9-node configurations. These configurations are recommended for production environments. For testing purposes, you can choose a smaller configuration.

In this section, the following topics are described:

- ▶ Requirements for a 3-node configuration
- ▶ Requirements for a 7-node configuration
- ▶ Requirements for a 9-node configuration

Requirements for a 3-node configuration

Here is the example 3-nodes configuration that we deployed:

- ▶ Number of servers used: Three virtual machines (VMs) (servers can be physical machines or VMs)
- ▶ Installed operating system: Red Hat Enterprise Linux 7.5 for POWER9.

Table 2-6 Three-node configuration

Node	CPUs	RAM	Storage
Master node 1	Eight	126 GB	One 1.8 TB solid-state drive (SSD)
Master node 2	Eight	126 GB	One 1.8 TB SDD
Master node 3	Eight	126 GB	One 1.8 TB SDD

Figure 2-2 and Figure 2-3 show the output of one of the VMs' storage space and CPUs.

```
[root@p460a26 ~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda       252:0   0  1.8T  0 disk
..vda1    252:1   0   10M  0 part
..vda2    252:2   0   1G  0 part /boot
..vda3    252:3   0  1.2T  0 part
..rhel_p460a26-root 253:0   0 100G  0 lvm /
..rhel_p460a26-swap 253:1   0   4G  0 lvm [SWAP]
..rhel_p460a26-var  253:2   0 100G  0 lvm /var
..rhel_p460a26-home 253:3   0   50G  0 lvm /home
..rhel_p460a26-ibm  253:4   0 500G  0 lvm /ibm
..rhel_p460a26-data 253:5   0 500G  0 lvm /data
..vda4    252:4   0 533.5G 0 part
..rhel_p460a26-docker 253:6   0 200G  0 lvm
```

Figure 2-2 Disk space of a VM

```
[root@p460a26 ~]# lscpu
Architecture:          ppc64le
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:  0-7
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):             8
NUMA node(s):          1
Model:                 2.0 (pvr 004d 0200)
Model name:            POWER8 (architected), altivec supported
Hypervisor vendor:    KVM
Virtualization type:  para
L1d cache:             64K
L1i cache:             32K
NUMA node0 CPU(s):    0-7
```

Figure 2-3 CPU output of a VM

Requirements for a 7-node configuration

When considering a 7-node configuration, consider the following items:

- ▶ The installation requires at least 10 GB on the root partition.
- ▶ If you plan to place /var on its own partition, reserve at least 10 GB for that partition.
- ▶ IBM Statistical Package for the Social Sciences (IBM SPSS®) Modeler add-on requirement: If you plan to install the SPSS Modeler add-on, add 0.5 CPU and 8 GB of memory for each stream that you plan to create.
- ▶ All servers must be synchronized in time (ideally through NTP or Chrony).
- ▶ SSH between nodes should be enabled.
- ▶ YUM should not be already running.

Table 2-7 shows a sample configuration.

Table 2-7 Sample configuration of a 7-node cluster

Node type	Number of servers ^a	CPU	RAM	Disk space
Control or storage	Three	Eight cores	46 GB	Minimum 300 GB with Extended File System (XFS) format for installer files partition + minimum 500 GB with XFS format for data storage partition + minimum 200 GB of extra raw disk space for Docker.
Compute	Three cores	16	64 GB	Minimum 300 GB with XFS format for installer file partition + minimum 200 GB of extra raw disk space for Docker. If you add more cores, a total of 48 - 50 cores that are distributed across multiple nodes is recommended.
Deployment	One	16	64 GB	Minimum 300 GB with XFS format for installer file partition + minimum 200 GB of extra raw disk space for Docker. If you add more cores, a total of 48 - 50 cores that are distributed across multiple nodes is recommended.

a. Bare metal or VM

Requirements for a 9-node configuration

Table 2-8 shows the minimum requirements for a 9-node configuration.

Table 2-8 Nine-node configuration

Node type	Number of servers ^a	CPU	RAM	Disk space
Control	Three	Four cores	16 GB	Minimum 500 GB with XFS format for installer files partition.
Storage	Three	Eight cores	48 GB	Minimum 500 GB with XFS format for installer files partition + minimum 500 GB with XFS format for data storage partition.
Compute	Three	16 cores	64 GB	Minimum 500 GB with XFS format for installer file partition. If you add more cores, a total of 48 - 50 cores that are distributed across multiple nodes is recommended.

a. Bare metal or VM

2.2.3 Configuring an HDP system

Figure 2-4 shows some recommendations for HDP deployment with IBM Watson Studio Local and IBM Watson Machine Learning Accelerator for a production system.

	System Mgmt Node	Master Node	Edge Node	Worker Node		
Cluster Type	All	All	All	Balanced	Performance	Server Dense
Server Model	1U LC921	1U LC921	1U LC921	2U LC922	2U LC922	1U LC921
# Servers (Min/Default/Max)	1 / 1 / 1	3 / 3 / Any	1 / 1 / Any	4 / 8 / Any	4 / 8 / Any	4 / 8 / Any
Sockets	2	2	2	2	2	2
Cores (total)	32	40	40	44	44	40
Memory	32GB	256GB	256GB	256GB	512GB	256GB
Storage - HDD (front)	2x 4TB HDD	4x 4TB HDD	4x 4TB HDD	4x 4TB HDD	4x 4TB HDD	
Storage - SSD (front)					4x 3.8TB SSD	
Storage - HDD (rear for OS)				2x 1.2TB HDD	2x 1.2TB HDD	
Storage Controller	MicroSemi PM8069 (internal)					
Network [*] - 1 GbE	Internal (4 ports OS)					
Cables [*] - 1 GbE	3 (2 OS + 1 BMC)					
Network ^{**} - 10 GbE	1x 2-port Intel (2 ports)	1x 2-port Intel (2 ports)	2x 2-port Intel (4 ports)	1x 2-port Intel (2 ports)	1x 2-port Intel (2 ports)	1x 2-port Intel (2 ports)
Cables ^{**} - 10 GbE	2 cables (DACs)	2 cables (DACs)	4 cables (DACs)	2 cables (DACs)	2 cables (DACs)	2 cables (DACs)
Operating System	RHEL 7.5 for P9					

* The 1 GbE network infrastructure hosts the following logical networks: campus, management, provisioning and service networks. See Section 7.4.1 for details.
** The 10GbE network infrastructure hosts the data network.

Figure 2-4 HDP recommended configurations

2.2.4 Configuring a proof of concept

Table 2-9 shows a proof of concept (PoC) configuration for a minimally functioning environment with cost-sensitive variations.

Table 2-9 Proof of concept configuration

Item	System management node	Master/edge node	Worker node
Cluster type	All	All	PoC
Server model	1U Power LC921server	1U Power LC921 server	1U Power LC922 server
Servers	One	One	Three
Sockets	Two	Two	Two
Cores	32	40	44
Memory	32 GB	256 GB	256 GB
Storage	Two 4 TB HDDs	Four 4 TB HDDs	Four 4 TB HDDs
Storage controller	MicroSemi PM8069 (internal)	MicroSemi PM8069 (internal)	MicroSemi PM8069 (internal)
Network ^a - 1 GbE	Internal (4-port OS)	Internal (4-port OS)	Internal (4-port OS)

Item	System management node	Master/edge node	Worker node
Cables ^a - 1 GbE	Three (two OSes + one baseboard management controller (BMC))	Three (two OSes + one BMC)	Three (Two OS + one BMC)
Network ^b - 10 GbE	One 2-port Intel (two ports)	Two 2-port Intel (four ports)	One 2-port Intel (two ports)
Cables ^b - 1 GbE	Two cables (direct-access cables (DACs))	Four cables (DACs)	Two cables (DACs)
Operating system	RHEL 7.5 for POWER9 processor-based systems	RHEL 7.5 for POWER9 processor-based systems	RHEL 7.5 for POWER processor-based systems

- a. The 1 GbE network infrastructure hosts the following logical networks: campus, management, provisioning, and service network.
- b. The 10 GbE network infrastructure hosts the data network.

2.2.5 Conclusion

IBM Watson Studio Local, IBM Watson Machine Learning Accelerator, and HDP deployed on IBM Power Systems servers, with its many hardware threads, high memory bandwidth, and tightly integrated NVIDIA GPUs, are suitable for running machine learning (ML) and DL computations that use open source frameworks on large data sets in enterprise environments.

2.3 Deployment options

In this section, we consider some of the deployment options for the IBM AI solutions and H2O Driverless AI.

2.3.1 Deploying IBM Watson Studio Local in stand-alone mode or with IBM Watson Machine Learning Accelerator

When IBM Watson Studio Local is installed as a stand-alone product, it provides a premier enterprise development and deployment environment for data scientists. Projects can be used to segment lines of business (LOBs) and use cases. Its integration with a datalake through the Hadoop Integration service is superb. You can use its environment for Jupyter with Python 3.6 and IBM PowerAI V1.5.3 for GPU to leverage the Power Systems nodes that have GPUs.

IBM Watson Machine Learning Accelerator is installed on POWER8 or POWER9 processor-based nodes with GPUs. Version 1.1.2 includes IBM PowerAI V1.5.4, which includes recent releases of DL frameworks like TensorFlow 1.12 with Keras.

IBM Watson Machine Learning Accelerator also supports multiple LOBs and use cases through IBM Spectrum Conductor consumers. Consumers can have different resources that are allocated to them and sharing policies can be defined statically or dynamically to give administrators full control over how the cluster is used. IBM Watson Machine Learning Accelerator Deep Learning Impact can help data scientists reach their goals quicker.

IBM Watson Studio Local can complement an IBM Watson Machine Learning Accelerator cluster by submitting remote jobs to run on it. The IBM Watson Studio Local notebooks can use Sparkmagic to connect to a Livy service application that is installed on IBM Watson Machine Learning Accelerator and submit Spark workloads on the SIG that is associated with the Livy application.

IBM PowerAI, IBM Watson Machine Learning Accelerator, and IBM Watson Studio Local can be installed in an IBM Cloud Private environment. That option enables resource sharing by all three products.

2.3.2 Using the Hadoop Integration service versus using an Apache Livy connector

IBM Watson Studio Local can connect to a Hadoop cluster by using an Apache Livy connection or the new Hadoop Integration service. The Hadoop Integration service has many advantages:

- ▶ You can push customized environments that are running in IBM Watson Studio Local to HDP.
- ▶ You can select the pushed environment in which notebooks connecting to HDP run.
- ▶ On HDP servers that are secured with Kerberos, use the data connector so that jobs may run as the calling user without needing a keytab for each user, which is important for fine-grained authorization and the associated logging of resource usage.
- ▶ You can define HDFS and Hive data sources.
- ▶ You can shape and transform data in HDP by using the IBM Watson Studio Local Refinery.

Using the IBM Watson Studio Local Hadoop Integration service to connect to HDP and Cloudera datalakes is a best practice.

2.3.3 Deploying H2O Driverless AI in stand-alone mode or within IBM Watson Machine Learning Accelerator

H2O Driverless AI can be installed as a stand-alone product or as an IBM Watson Machine Learning Accelerator application. If you install H2O Driverless AI as an IBM Watson Machine Learning Accelerator application, you can manage it (start and stop) from the IBM Watson Machine Learning Accelerator console.

To install H2O Driverless AI as an application on IBM Watson Machine Learning Accelerator, download the Linux tar.sh file at the following website:

<https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/dai/rel-1.3.1-12/ppc64le-centos7/dai-1.3.1-linux-ppc64le.sh>

The H2O Driverless AI software is available for use in pure user-mode environments as a self-extracting tar.sh archive. This form of installation does not require a privileged user to install or to run.

For more information about the H2O.ai tar.sh package, see [Using Driveless AI 1.5.4](#).

The IBM NVLink interface enables the H2O Driverless AI next-generation AI platform to get the maximum performance gains. Together, H2O Driverless AI and IBM PowerAI provide companies with a data science platform or an *AI workbench* that addresses a broad set of use cases for ML and DL in every industry.

These integrations happen mainly with H2O Driverless AI, but there is a possibility to integrate with H2O Sparkling Water as well.

H2O Driverless AI

H2O Driverless AI provides the following functions:

- ▶ Automates data science and ML workflows.
- ▶ H2O Driverless AI is started on a single host that can have either GPUs or run with CPUs.
- ▶ Shared file system for data and logs.
- ▶ If you use GPUs, the entire host is taken (with current integration).
- ▶ An application instance is created for each user of H2O Driverless AI.
- ▶ Environment variables through parameters are used to configure H2O Driverless AI.
- ▶ Fails over to another host if H2O Driverless AI goes down, and IBM Spectrum Conductor starts it on another host.

Figure 2-5 shows H2O Driverless AI on the management console.

The screenshot shows the IBM Spectrum Computing Cluster Management Console interface. The top navigation bar includes 'Workload', 'Resources', 'System & Services', and 'Reports & Logs'. The main area displays 'Application Instances in All Consumers' with a table showing one instance: 'H2O-DriverlessAI' (Started, Top Consumer: /DAI). Below this, a detailed view for 'Application Instance: H2O-DriverlessAI' is shown, with tabs for 'Summary', 'Services', 'Service Groups', 'Dependencies', 'Performance', 'Hosts', and 'Packages'. The 'Summary' tab is selected, displaying details such as Application Instance: H2O-DriverlessAI, State: Started, Service Summary, Registered Time: 10/10/2018, 23:31:11 GMT+2, Start Time: 10/24/2018, 21:08:48 GMT+2, Up Time: 00:09:47, UUID: 60f72c62-5dc6-42e9-91bd-4f70d8509eab, Description: H2O Driverless AI, Last Deploy Time: -, Last Stop Time: 10/24/2018, 21:06:36 GMT+2, Top Consumer: /DAI, and New Consumers: /DAI/H2O-DriverlessAI-DAItest. The 'Outputs' section shows the URL: <http://node23.conductor.iccmop:12345>.

Figure 2-5 H2O Driverless AI as shown in the management console

H2O Sparkling Water

H2O Sparkling Water provides the following functions:

- ▶ You can use it to combine the ML algorithms of H2O Driverless AI with the capabilities of Spark.
- ▶ It runs as a notebook in a SIG.
- ▶ When the notebook starts, it forms a mini-cluster of executors.
- ▶ These executors stay alive for the entire duration of the notebook.
- ▶ IBM Spectrum Conductor disables preemption to prevent reclamation of these hosts.
- ▶ Multiple users can share an H2O Sparkling Water notebook instance or have dedicated ones per user.

If you want to work with H2O Driverless AI in IBM Watson Studio Local, install the H2O Flow add-on to use an H2O Flow session within IBM Watson Studio Local to create documents and work with models.

For more information, see [H2O Flow add-on](#).

IBM Spectrum Conductor integration with H2O Driverless AI

At the time of writing, H2O Driverless AI does not support NVIDIA CUDA 10.0, which is a prerequisite for IBM Watson Machine Learning Accelerator V1.1.2 and IBM PowerAI V1.5.4. The workaround is to install the NVIDIA CUDA 9.0 libraries without the driver. The IBM Watson Machine Learning Accelerator cluster continues to run with the CUDA 10.0 driver, but CUDA 9.0 libraries are available for H2O Driverless AI.

To configure the workaround, complete the following steps:

1. Run the following command:

```
wget  
https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/cuda-repo-rhe17-9-0-local-9.0.176-1.ppc64le-rpm
```

2. Run the following command:

```
rpm -i cuda-repo-rhe17-9-0-local-9.0.176-1.ppc64le.rpm
```

3. Run the following command:

```
yum install cuda-toolkit-9-0
```

4. Install H2O Driverless AI by completing the following steps:

- a. Download the Linux tar.sh package from [the H2O website](#).
- b. Download conductor-h2o-driverlessai-master.zip from [GitHub](#).
- c. Install H2O Driverless AI on all the nodes by reviewing the following video at [Box](#).
- d. Run `cd /var/dai`.
- e. Install `dai.sh`.
- f. Extract the GitHub package to the same directory. Make sure that the directory has the same owner and group of the executor user, for example, `egoadmin`.

5. After installing H2O Driverless AI, edit or create the `/etc/dai/EnvironmentFile.conf` file to contain the following parameter definitions:

```
DRIVERLESS_AI_CUDA_VERSION=cuda-9.0  
LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64:/usr/local/cuda-9.0/extras/CUPTI/lib64
```

6. Log in to the IBM Spectrum Conduction console to register H2O Driverless AI as an application instance that can be stopped and started by using the application management support. Starting at the console, complete the following steps:
 - a. Click **Workload** and select the application instance.
 - b. Click **Add**.
 - c. Register by using the yaml file by selecting the yaml file and completing the form.
 - d. Specify the top-level consumer.
 - e. Specify the resource group (RG).
 - f. Specify the data directory, for example, /var/dai-data.
 - g. Specify the execution user, which is the OS user running the application, which is the same as the SIG execution user.
 - h. Specify the installation directory, for example, /var/dai/dai....
 - i. Click **Register**.
7. Select the H2O Driverless AI application instance and start it.
8. Review the output section for **dai_url**.
9. Click **dai_url** to start H2O Driverless AI.

2.3.4 Running Spark jobs

In an environment with IBM Watson Machine Learning Accelerator, IBM Watson Studio Local, and HDP, the preferred place to run Spark jobs depends on several factors. When the data of interest is in the HDP datalake, running Spark jobs on the datalake takes advantage of the data locality and distributed nature of Spark.

If you want to take advantage of GPUs and the IBM PowerAI optimized DL frameworks, run the jobs on IBM Watson Machine Learning Accelerator or IBM Watson Studio Local (assuming IBM Watson Studio Local has Power Systems servers with GPUs).

2.4 IBM Watson Machine Learning Accelerator and Hortonworks Data Platform

The following main components are included with IBM Watson Machine Learning Accelerator:

- ▶ IBM Spectrum Conductor Deep Learning Impact:
 - Data Manager and extract, transform, and load (ETL)
 - Training Visualization and Training
 - Hyper-parameter optimization
- ▶ IBM Spectrum Conductor:
 - Multi-tenancy support and security
 - User reporting and charge back
 - Dynamic resource allocation
 - External data connectors

Officially, IBM Watson Machine Learning Accelerator with IBM Spectrum Conductor integrates with IBM Cloud Private and IBM Watson Studio. Through this integration, it runs notebooks from SIGs to submit Spark workloads.

However, IBM Watson Machine Learning Accelerator can easily integrate with HDP in two ways:

- ▶ You can configure data connectors that enable HDFS data to be accessed by IBM Watson Machine Learning Accelerator models.
- ▶ IBM Watson Machine Learning Accelerator can submit a Spark job to an HDP cluster that is configured with Apache Livy. Running the job directly on HDP takes advantage of the data locality.

For both approaches, you can request them directly from a notebook by using the following sample code:

- ▶ Livy

```
%load_ext sparkmagic.magics
```

To see all the available options for the sparkmagic extension inside the notebook, run the following command:

```
help as: spark?  
%load_ext sparkmagic.magics  
%spark add -s <session> -l python -u <hdp-livy_URL> -a u -k
```

The Livy UI provides full details about each interaction inside the session, including input and output returned.

- ▶ HDFS

```
df =  
spark.read.load("hdfs://<hdp-domain_url>:8020/user/user1/datasets/cars.csv",  
format="csv",)
```

2.5 IBM Watson Studio Local with Hortonworks Data Platform

There are two types of integration for IBM Watson Studio Local with Hadoop:

- ▶ Hadoop is used as a data source. IBM Watson Studio Local supports both secure and non-secure connections to HDFS and Hive. You can configure a connection by using the UI or programmatically. If you configure a connection by using the UI, you can browse for files, preview them, and share data sources with project collaborators.
- ▶ Using a Spark environment in the Hadoop cluster to run IBM Watson Studio Local notebooks and batch jobs. The main advantage of this approach is cutting down the performance impact of moving data from Hadoop to the Spark cluster in IBM Watson Studio Local.

Figure 2-6 shows the logical connections between IBM Watson Studio Local and Hortonworks.

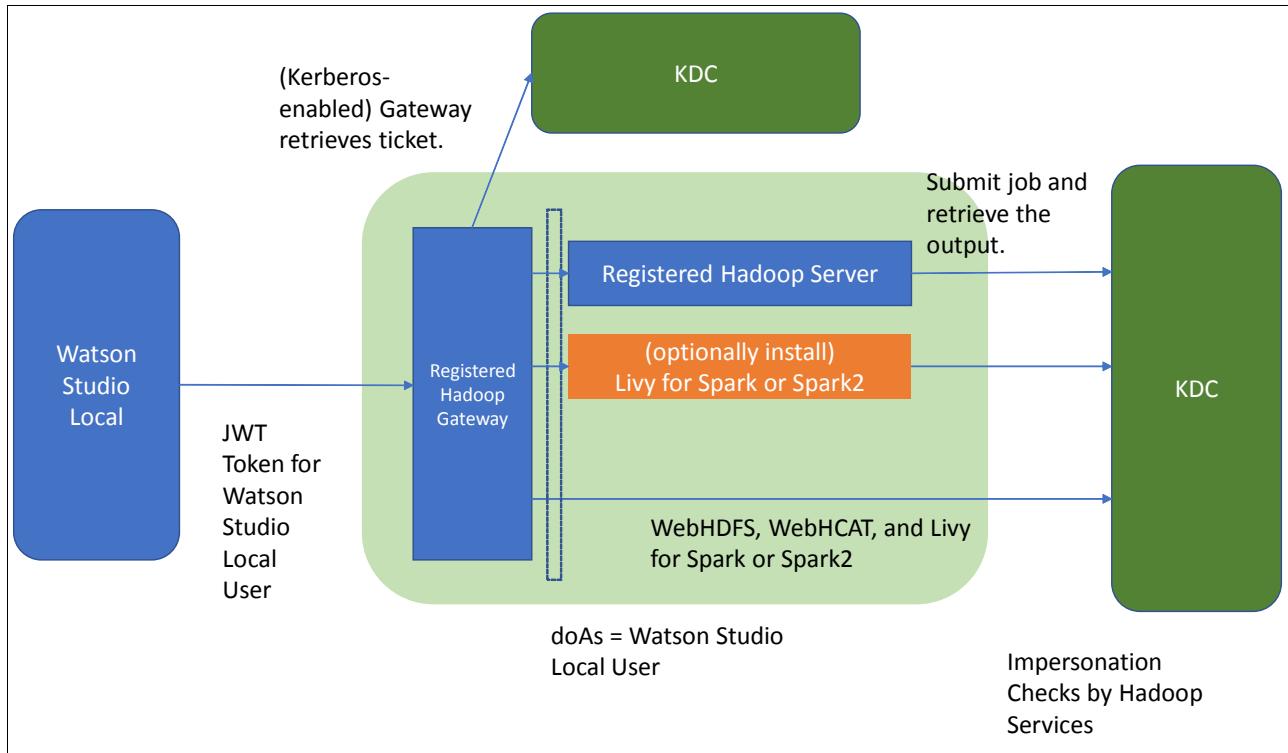


Figure 2-6 Watson Studio connections with Hortonworks

IBM Watson Studio Local interacts with an HDP cluster through four services:

- ▶ **WebHDFS**
WebHDFS is used to browse and preview HDFS data.
- ▶ **WebHCAT**
WebHCAT is used to browse and preview Hive tables.
- ▶ **Livy for Spark**
- ▶ **Livy for Spark2**
Livy for Spark and Livy for Spark2 are used to submit jobs to Spark or Spark2 engines on the Hadoop cluster.

Note: WebHCAT is no longer supported as of HDP 3.0. IBM Watson Studio Local integrates with HDP through HDFS and Spark integrates through Livy as of HDP 3.0.

The Hadoop registration service should be installed on an edge node of the HDP cluster. The gateway component authenticates all incoming requests and forwards them to the Hadoop services. In a Kerberized cluster, the keytab of the Hadoop registration service user and the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) keytab for the edge node are used to acquire the ticket to communicate with the Hadoop services. All requests to the Hadoop service are submitted as the IBM Watson Studio Local user.

For more information, see [Set up a remote Hadoop cluster to work with Watson Studio Local](#).

2.6 IBM Watson Studio Local with IBM Watson Machine Learning Accelerator

When IBM Watson Studio Local is installed on Power Systems servers with GPUs (such as the Power S822LC HPC or Power AC922 servers), the IBM PowerAI DL libraries are automatically included (including GPU-enabled libraries for TensorFlow and Caffe). Users of IBM Watson Studio Local can create notebooks and build models that use these frameworks and the Power Systems and GPU-optimized libraries.

These IBM PowerAI libraries that are included with IBM Watson Studio Local come without support, which is similar to how the IBM PowerAI base product is available for no charge without support. Clients who want support can purchase IBM Watson Machine Learning Accelerator, which also comes with capabilities such as IBM Spectrum Conductor Deep Learning Impact. IBM Spectrum Conductor Deep Learning Impact provides tuning for DL model training tasks, and IBM Spectrum Conductor has multi-tenant Spark cluster capabilities. IBM Watson Studio Local can be integrated with IBM Watson Machine Learning Accelerator by using the Apache Livy interface. In this scenario, IBM Watson Studio Local is the development and collaboration platform, and the IBM Watson Machine Learning Accelerator cluster is used to run model training.

In general terms, IBM PowerAI integrated with IBM Watson Studio Local provides optimized DL frameworks and libraries to leverage IBM Power Servers with GPUs (the Power AC922 server). As mentioned earlier, IBM Watson Studio Local is the platform where notebooks are created, data sources are identified and accessed, and collaboration between team members is enabled. IBM Watson Machine Learning Accelerator, if available, can be used for Spark-based ETL and DL model tuning and training. IBM Watson Studio Local is where the user creates the notebook (where the model training program is created). Without IBM Watson Machine Learning Accelerator, the training job is run on the compute cluster that is part of IBM Watson Studio Local. With IBM Watson Machine Learning Accelerator, the training job is scheduled on the IBM Watson Machine Learning Accelerator cluster.

Both IBM Watson Studio Local and IBM PowerAI are enterprise software offerings from IBM for data scientists that are built with open source components. IBM Watson Studio Local provides interactive interfaces such as notebooks and RStudio. IBM PowerAI provides DL frameworks such as TensorFlow and Caffe, among others, which are built and optimized for IBM Power Systems servers.

Although IBM PowerAI libraries are included with IBM Watson Studio Local and automatically deployed when IBM Watson Studio Local is installed, there is no customer support that is included for IBM PowerAI. To get support (in addition to extra capabilities), clients can purchase and deploy IBM Watson Machine Learning Accelerator instead. An example of how a client can leverage both of these offerings together is where data scientists can collaborate, experiment, and build their models in IBM Watson Studio Local and run the production training (with multiple GPUs) on a IBM Watson Machine Learning Accelerator cluster.

The Power AC922 server is the best one for enterprise AI. POWER processor-based servers have a number of significant advantages over Intel -based servers in performance and price. Work with your colleagues in the Analytics organization to position Power Systems as the preferred platform for the client's IBM Watson Studio Local deployment and their data science initiatives.

IBM Watson Studio Local is available as part of the IBM enterprise private cloud offering that is called IBM Cloud Private.

2.7 IBM Spectrum Scale and Hadoop Integration

The IBM Spectrum Scale HDFS Transparency Connector is a remote procedure call (RPC) API that connects IBM GPFS with HDFS. It offers a set of interfaces that enable applications to use the HDFS Client to access the IBM Spectrum Scale namespace by using HDFS traditional requests.

Figure 2-7 shows the flow of the data from big data applications to the IBM Spectrum Scale namespace.

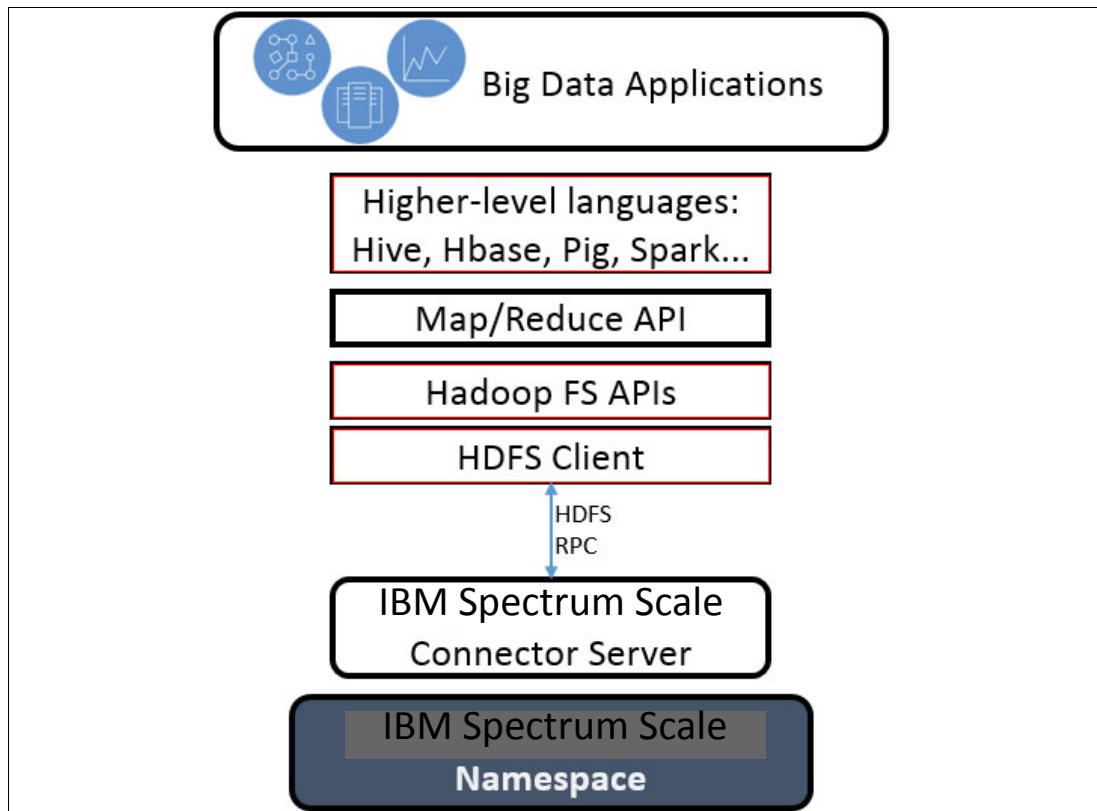


Figure 2-7 IBM Spectrum Scale flow of data architecture

All data transmission and metadata operations in HDFS are performed through RPCs and processed by NameNode and DataNode services within HDFS. IBM Spectrum Scale HDFS Transparency Connector integrates both NameNode and DataNode services and responds to the request like HDFS does. IBM Spectrum Scale HDFS Transparency Connector ensures that any application that attempts to read, write, or run any operation on the HDFS receives the expected response.

Some advantages of HDFS transparency are:

- ▶ HDFS-compliant APIs and shell-interface command-line interface (CLI).
- ▶ Application client isolation from storage. The application client may access the IBM Spectrum Scale file system without the IBM Spectrum Scale Client installed.
- ▶ Improved security management by using Kerberos authentication and encryption in RPC.
- ▶ File system monitoring by Hadoop Metrics2 integration.
- ▶ IBM Spectrum Scale services management directly on the Ambari GUI.

Avoiding unnecessary data moves reduces cost in a Hadoop environment like HDP. With IBM Spectrum Scale, HDP components may access a namespace (file system mount point) on the Hadoop cluster by using the same data that is accessible to any other software (SAS, IBM Db2, Oracle Database, SAP, and others) without using a command such as **`distcp`**, which is a common problem when you run HDFS and must access data from an external file system.

2.7.1 Information Lifecycle Management

IBM Spectrum Scale uses the policy-based Information Lifecycle Management (ILM) toolkit to automate the storage of data into tiered storage. With the ILM toolkit, you can manage file placement rules across different pools of storage, like flash drives for hot data, SATA disks for commonly accessed data, and even cold data on the cloud or a tape driver.

With IBM Spectrum Scale policy-based ILM tools, you can accomplish the following tasks:

- ▶ Create storage pools to partition a file system's storage into collections of disks or a RAID with similar properties.
- ▶ Create file sets to partition the file system namespace to allow administrative operations at a finer granularity than that of the entire file system.
- ▶ Create policy rules based on data attributes to determine the initial file data placement and manage file data placement throughout the life of the file.

Some rules conditions that can be used are:

- ▶ Date and time when the file was last accessed.
- ▶ Date and time when the file was last modified.
- ▶ File set name (directory).
- ▶ File name or extension.
- ▶ File size.
- ▶ Owner of the file (user ID and group ID).

You can create and manage policies and policy rules by using the IBM Spectrum Scale CLI or the GUI. To create a policy rule by using the GUI, complete the following steps:

1. Log in to the IBM Spectrum Scale web interface and in the left pane select **Files** → **Information Lifecycle**. Click **Add Rule**. The window that is shown in Figure 2-8 opens.

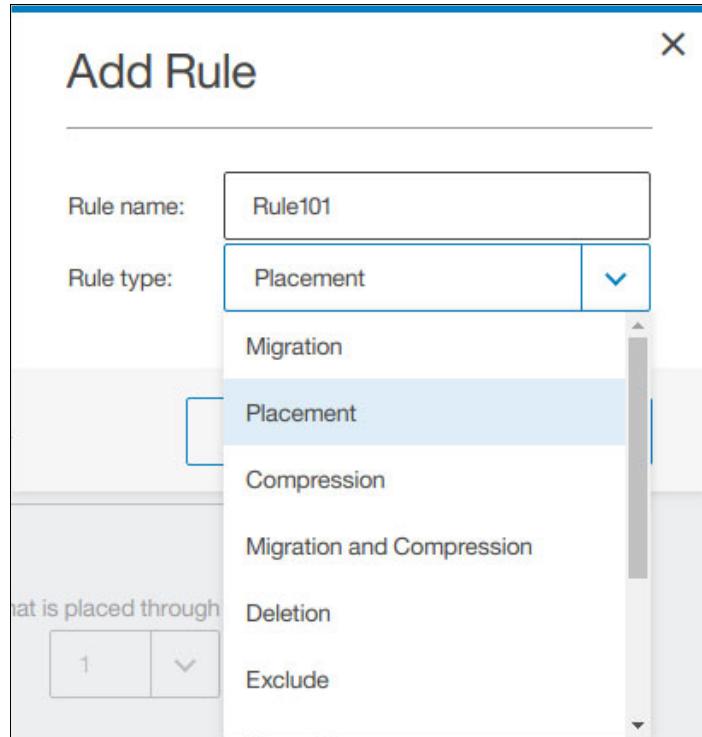


Figure 2-8 IBM Spectrum Scale GUI: Information Lifecycle Add Rule window

2. Select the type of rule that you want to create, such as Placement, Migration, Compression, or Deletion. The window that is shown in Figure 2-9 opens.

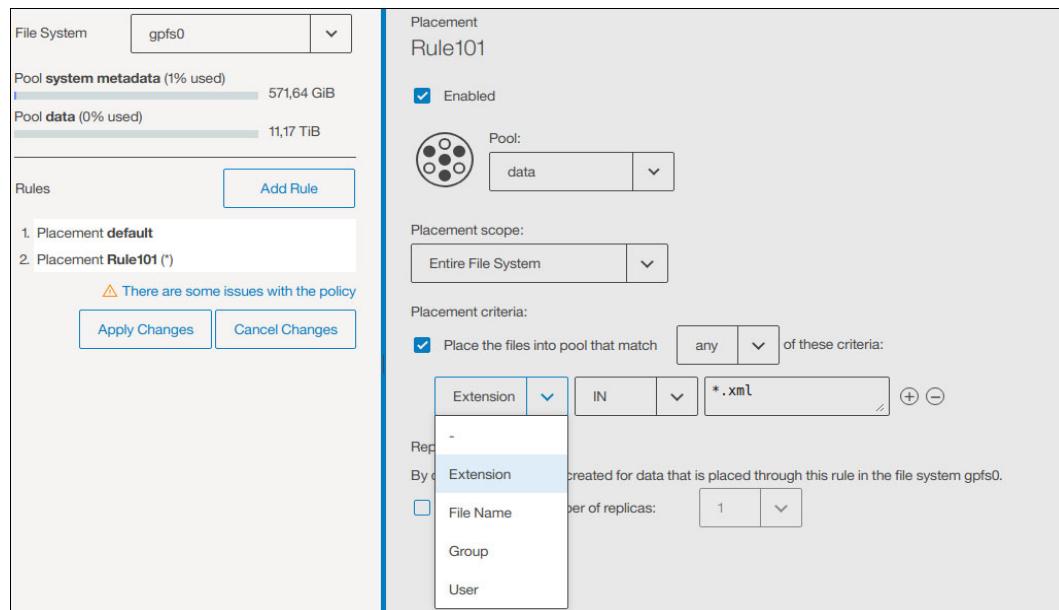


Figure 2-9 IBM Spectrum Scale GUI: Information Lifecycle Rule Type window

- Select the pool where the new files must be placed (when you are creating a placement policy), the Placement Criteria (that is, only files with the extension .xml), and more.

For a Migration Policy rule, you can select the source and target pools where the files must be moved, and then the Rules to trigger the migration. An example is when the source pool achieves a defined utilization threshold or a file is not accessed for a number of days.

Figure 2-10 shows some more configurations that can be selected for the Information Lifecycle rule.

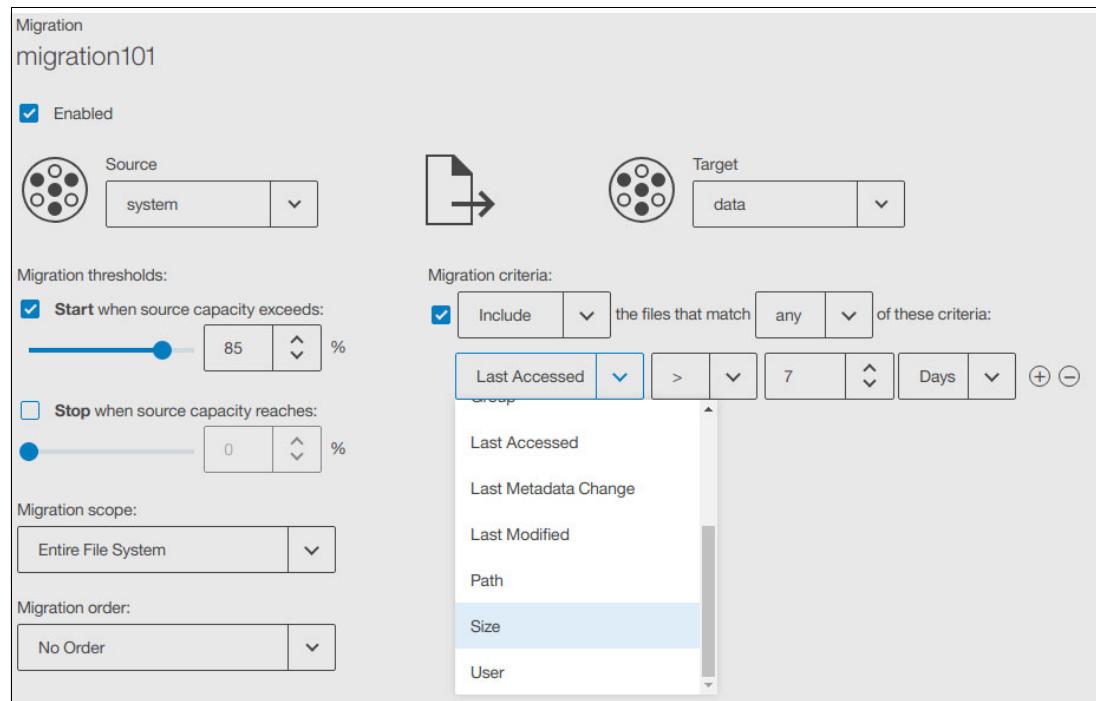


Figure 2-10 IBM Spectrum Scale: Information lifecycle rules configuration

There are many types of ILM rules that can be created on IBM Spectrum Scale. The GUI makes creating a policy to meet your needs simple and fast.

2.8 Security

Security is at the forefront of most clients' minds. Security incidents and data breaches often make significant news because of the potential impact they have on the business and their customers. Although some companies are making a move to the public cloud, others want to keep their data local and secure behind their firewall.

Protecting the datalake is paramount in an enterprise. The security mechanisms must include authentication, authorization, audit, encryption, and policy management:

- ▶ Authentication is the process of proving who you are.
- ▶ Authorization is the process of verifying that you have the authority to access a resource.
- ▶ Records that indicate who did what provide the audit capability.
- ▶ Encryption is the mechanism that is used to protect data-at-rest and in motion.
- ▶ Policy management is done through the product administration consoles.

Ideally, these mechanisms can be managed in a consistent and cohesive fashion.

2.8.1 Datalake security

Hortonworks uses Apache Ranger for its centralized security management, although Ambari is used for some of the fundamental setup, such as enabling Kerberos. Apache Ranger is a framework and service to enable, monitor, and manage security for the Hadoop platform and services. The Administration Portal is the Ranger interface for security administration. Hadoop components like Knox, YARN, Hive, HBase, and HDFS have lightweight plug-ins that provide integration with Ranger.

From the Ambari administration console, administrators can enable and configure Kerberos for authentication and set perimeter security policies by using Knox.

From the Ranger administration console:

- ▶ Administrators can set fine-grained access control for the authorization policies.
- ▶ Administrators can view the centralized audit reports showing user activity.
- ▶ Administrators can configure Ranger KMS for cryptographic key management for HDFS Transparent Encryption. It lets you create and manage the keys, in addition to audit and access control of the keys. You cannot use to enable directly HDFS encryption.

Hadoop authentication

Hadoop uses Kerberos for authentication and the propagation of identities for users and services. Kerberos includes the client, server, and trusted party known as the Kerberos Key Distribution Center (KDC). The KDC is a separate server from the Hadoop cluster. It includes a database of users and services that are known as *principals*, and is composed of an Authentication Server and a Ticket Granting Service (TGS). The Authentication Server is used for the initial authentication. It issues a Ticket Granting Ticket (TGT) to the authenticated principal. The TGS is used to get service tickets by using the TGT. Host and service resources use a special file that is called a *keytab* that includes their principal and key. The keytab resolves the issue of providing a password when decrypting a TGT. The Hortonworks security documentation provides the detailed steps for enabling Kerberos, including the Kerberos server setup.

If you already have a Lightweight Directory Access Protocol (LDAP) server solution, you can use configure various HDP components such as Ambari, Knox, and Ranger to use it. IBM Watson Studio Local also supports the configuration of an external LDAP server for its users. The two solutions can share an LDAP server and avoid the problem of defining users in multiple places.

The Apache Knox Gateway is a reverse proxy that provides the perimeter security for a Hadoop cluster. It exposes Hadoop REST and HTTP services for HDFS, Hive, HBase, and others without revealing the cluster internals. It provides SSL for over the wire encryption and encapsulates the Kerberos authentication.

Hadoop authorization

The Apache Ranger service can be installed through the Ambari Add Service wizard. Before adding Ranger, set up a database instance and an Apache Solr instance. The database is used for administration and logging. Ranger uses Apache Solr to store audit logs. After Ranger is installed, the Ranger plug-ins can be enabled for each of the services being administered, including HDFS, HBase, HiveServer2, Storm, Knox, YARN, and Kafka, and NiFi.

Log in to the Ranger portal at `http://ranger_host:6080` to open the Ranger Console. The Service Manager page opens and shows the Access Manager, Audit, and Settings tabs at the top of the page:

- ▶ The **Access Manager → Resource Based Policy** option opens the Service Manager page so that you can add access policies for the services that are plugged into Ranger.
- ▶ The **Access Manager → Tag Based Policies** option opens the Service Manager for Tag Based policies page so that you can add tag-based services that you use to control access to resources that access multiple Hadoop components.
- ▶ The **Access Manager → Reports** option opens the Reports page where you can generate user access reports.
- ▶ The **Settings → User/Groups** option shows a list of users and groups that can access the Ranger portal.
- ▶ The **Settings → Permissions** option opens a Permissions page where you can edit the permissions for users and groups.

Hadoop auditing

Ranger offers the ability to store audit logs by using Apache Solr. Solr is an open source highly scalable search platform. Store your audit logs in HDFS so that they can be exported to a security information and event management (SIEM) system. Having the audit log data in HDFS also provides the ability to create an anomaly detection application by using ML and DL algorithms.

Selecting the Ranger console **Audit** tab opens an Audit page where you can create a search to monitor a user's activity.

Hadoop data protection

Hadoop provides protection mechanisms for data-in-motion and data-at-rest.

Hadoop wire encryption

Data on the wire is encrypted to ensure its privacy and confidentiality. Hadoop can be configured to encrypt data as it moves into the cluster, moves through the cluster, and as it moves out of the cluster. There are a number of protocols that must be configured for encryption: RPC, data transfer protocol (DTP) HTTP, and Java Database Connectivity (JDBC).

Clients use RPC to interact directly with the Hadoop Cluster. DTP is used when the client is transferring data within a data node. Clients use HTTP to connect to the cluster through a browser or a REST API. HTTP is also used between mappers and reducers during a data shuffle. JDBC is used to communicate with the Hive server.

RPC encryption is enabled by setting the HDFS property `hadoop.rpc.protection=privacy`.

DTP encryption is enabled by setting the HDFS property `dfs.encrypt.data.transfer=true`.

The DTP encryption algorithm, “3des” or “rc4”, is set by using the `dfs.encrypt.data.transfer.algorithm` property.

For more information about enabling SSL encryption for the various Hadoop components, see [Enabling SSL for HDP Components](#).

HDFS Transparent Data Encryption

HDFS encryption is an end-to-end encryption mechanism for stored data in HDFS. The data is encrypted by the HDFS client during the write operation and decrypted by an HDFS client during a read operation. HDFS sees the data as a byte stream during these operations. HDFS encryption is composed of encryption keys, encryption zones, and Ranger KMS. The keys are a new level of permission-based access beyond the standard HDFS access control. The encryption creates a special HDFS directory so that all data in it is encrypted. The Ranger KMS generates and manages the encryption zone keys, provides the access to the keys, and stores audit data that is associated with the keys.

The native HDFS encryption feature is supported since HDFS Transparency 3.0.0.

2.8.2 IBM Watson Machine Learning Accelerator security with Hadoop

IBM Spectrum Conductor supports Kerberos authentication, which can be extended to HDFS. You can authenticate by using a Kerberos TGT or by using a principal with a keytab. The keytab is preferred for long-running applications.

The SIG to which you submit applications must reference a path to the Hadoop configuration. Set the SIG environment variable **HADOOP_CONF_DIR** to the path of the Hadoop configuration. For example:

```
HADOOP_CONF_DIR=/opt/hadoop-2-6-5/etc/hadoop.
```

When submitting a Spark batch job with a keytab for HDFS, specify the principal and keytab as options that are passed with the **--conf** flag. This **spark-submit** command shows the syntax for keytab authentication:

```
spark-submit --master spark://spark_master_url --conf
spark.yarn.keytab=path_to_keytab --conf spark.yarn.principal=principal@REALM.COM
--class main-class application.jar hdfs://namenode:9000/path/to/input
```

For access through Livy, follow the HDP instructions to configure the Livy service to authenticate with Kerberos. Those instructions include creating a user, principal, and keytab for Livy.

2.8.3 IBM Watson Studio Local security

When setting up HDP to work with IBM Watson Studio Local by using the Hadoop registration service, more setup is required for the Hadoop cluster that is configured with Kerberos. A gateway with JSON Web Token (JWT)-based authentication is configured so that IBM Watson Studio Local users can securely authenticate with the registration service.

Figure 2-11 shows the various components of an IBM Watson Studio Local and Hadoop Integration.

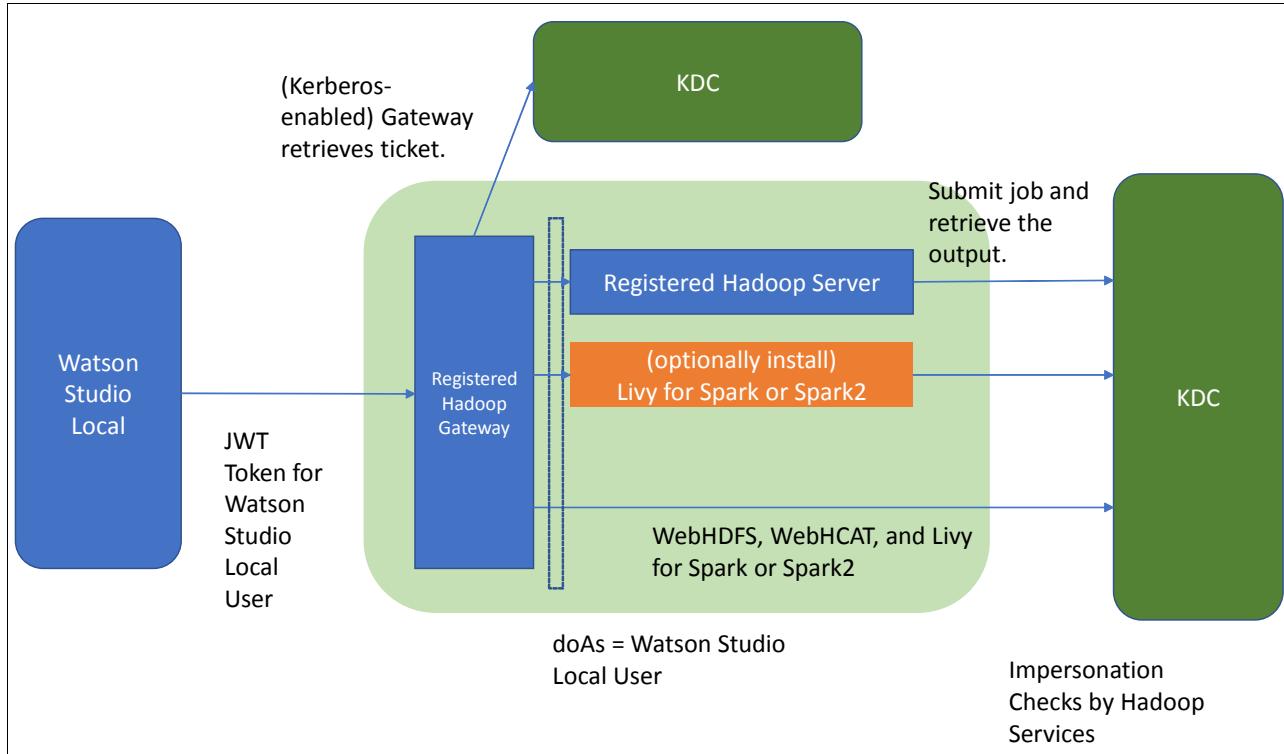


Figure 2-11 IBM Watson Studio Local Hadoop registration service

Use a Kerberos keytab for the Hadoop registration service user and an SPNEGO keytab for the edge node to acquire the ticket to communicate with the Hadoop services. The request that is submitted to the Hadoop services is submitted as the IBM Watson Studio Local user.

For access through Livy, follow the HDP instructions to configure the Livy service to authenticate with Kerberos. Those instructions include creating a user, principal, and keytab for Livy's use.

Here are some Watson Studio considerations to note when configuring your security:

- ▶ To help with General Data Protection Regulation (GDPR) readiness, install IBM Watson Studio Local Version 1.2.0.3 or later.
- ▶ Disk volumes should be encrypted. To encrypt data storage, use Linux Unified Key Setup-on-disk-format (LUKS). If you decide to use this approach, format the partition with the highly scalable Linux XFS before you install IBM Watson Studio Local.
- ▶ IBM Watson Studio Local can be accessed only through SSL/TLS (HTTPs). For more information, see 2.8, “Security” on page 50. Use JDBC/SSL-based mechanisms to communicate to remote data sources.
- ▶ Access failures are recorded in the logs.
- ▶ The Hadoop Integration service uses HTTPs, and enables secure communication.

2.8.4 IBM Spectrum Scale security

The IBM Spectrum Conductor software component and IBM Spectrum Scale wrap security features around these frameworks for production deployments in many regulated organizations. These solutions employ the current security protocols and are subjected to extensive security scanning and penetration testing. The IBM Spectrum Conductor software implements end-to-end security, from data acquisition and preparation to training and inference.

The product security implementation has the following features:

- ▶ Authentication: Support for Kerberos, Active Directory (AD) and LDAP, and operating system (OS) authentication, including Kerberos authentication for HDFS.
- ▶ Authorization: Fine-grained access control, access control list (ACL) or role-based control (RBAC), Spark binary lifecycle, notebook updates, deployments, resource plan, reporting, monitoring, log retrieval, and execution.
- ▶ Impersonation: Different tenants may define production execution users.
- ▶ Encryption: There is SSL and authentication between all daemons and storage encryption by using IBM Spectrum Scale.



Integrating new data

This chapter describes what the options are for integrating new data into the datalake and how it can be used to improve artificial intelligence (AI) models.

People are more connected, which leads to an accumulation of data sources and constantly growing data. The increased volume of data requires ever-increasing computing power to derive value from that data. The speed and directions from which data enters the enterprise is increasing due to advances in network technology, which results in data coming in faster than you can process it. The faster the data enters and the more varied the sources, the harder it is to derive value from the data. Fortunately, large data lakes and AI can help you with these issues.

The following topics are covered in this chapter:

- ▶ Data ingestion overview
- ▶ Types of data ingestion
- ▶ Options for data ingestion
- ▶ Using data connectors to work with external data sources
- ▶ How integration improves the artificial intelligence models

3.1 Data ingestion overview

Data ingestion is the process of bringing data into the data processing system, which in this case is Hortonworks Data Platform (HDP). The process consists of importing, transferring, loading, and processing data for later use or storage. It involves connecting to various data sources, extracting the data, and detecting changed data. Data ingestion subsystems must fetch data from various sources, such as IBM Db2, web logs, application logs, streaming data, social media, and many other sources.

Figure 3-1 shows how data ingestion works in HDP.

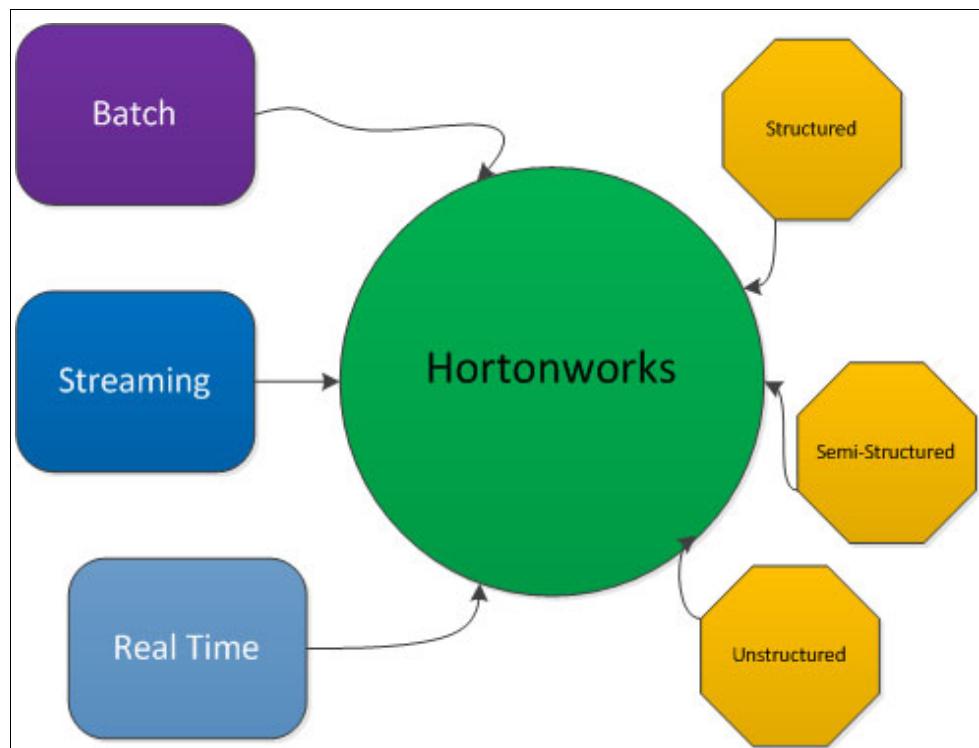


Figure 3-1 Data ingestion

Figure 3-2 shows the layout of HDP.

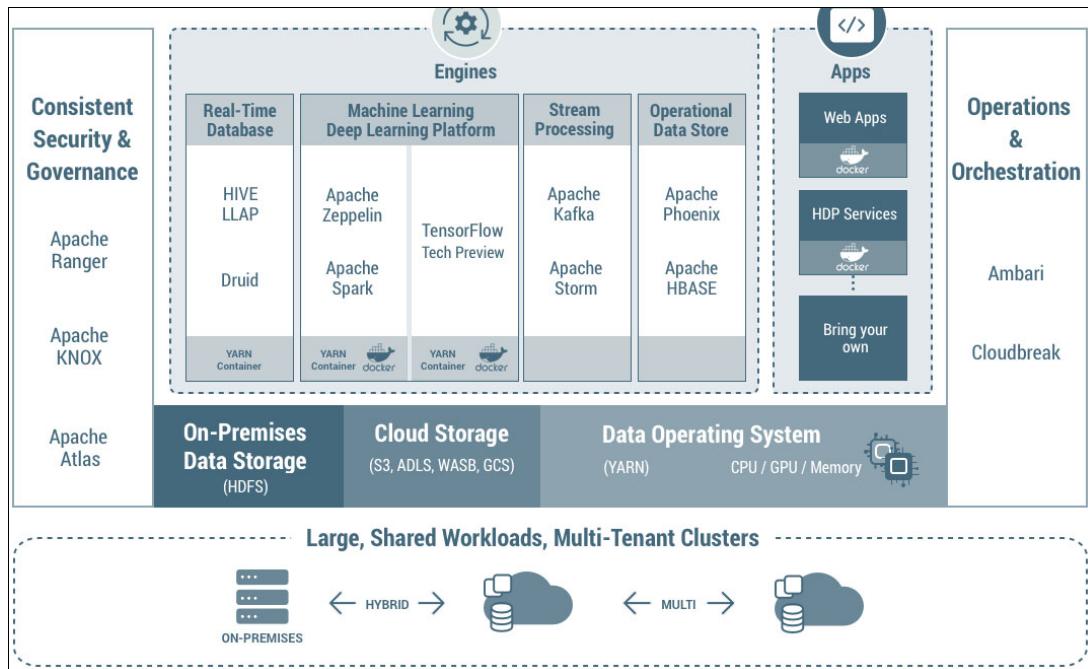


Figure 3-2 Hortonworks Data Platform

3.2 Types of data ingestion

There are two different platforms that Hortonworks supports: HDP and Hortonworks DataFlow (HDF). HDP is best known for handling data-at-rest, and HDF is known for its efficiency regarding data-in-motion. Both platforms are independent from each other and can operate separately, but are often integrated.

Some components of HDP are Hadoop Distributed File System (HDFS), YARN, MapReduce, Tez, Hive, HBase, Pig, Sqoop, and other useful tools.

Figure 3-3 shows an overview of HDP in the Ambari console.

Figure 3-3 Hortonworks Data Platform overview

In HDP, you can store any type of data, whether it is structured, semi-structured, or unstructured. HDP can store a large amount of data, that is, terabytes and petabytes of many different data types. It supports multiple standby nodes and enables clusters to scale up to thousands of nodes for high scalability and availability.

HDF solves the real-time challenges of collecting and transporting data from many sources and provides interactive command and control of live flows with full and automated data provenance that is powered by Apache NiFi, Apache Kafka, and Apache Storm to name a few.

Figure 3-4 shows an overview of the different types of data that can be ingested with HDF.

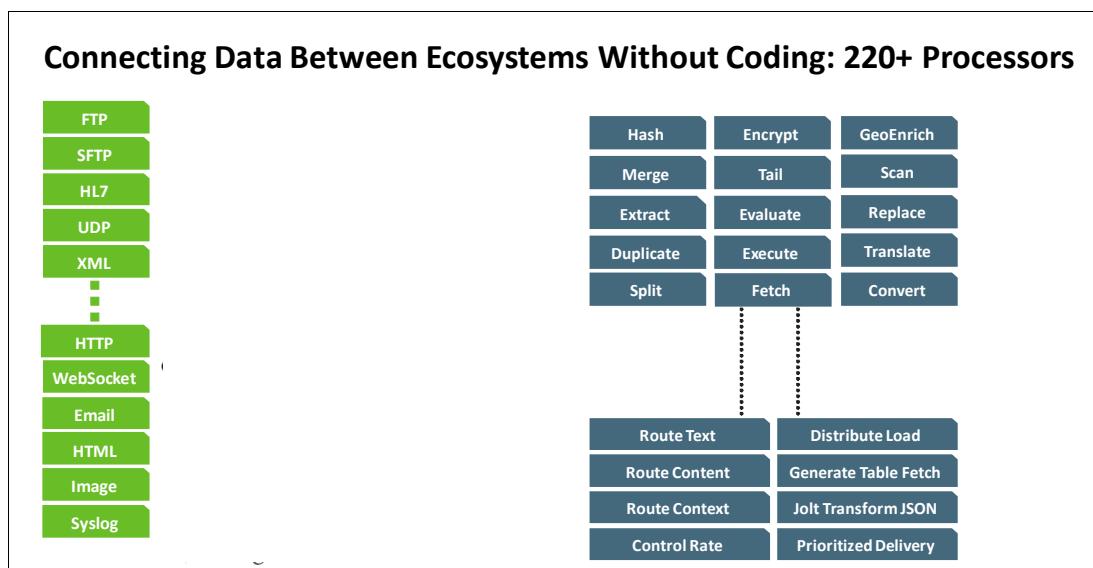


Figure 3-4 HDF data ingestion

But how exactly does new data get into your cluster? Data can be ingested into your cluster from many different sources, such as new log entries from your web server, new sensor data from your Internet of Things (IoT) devices, or new stock trades. Here is where Apache NiFi, Apache Kafka, and Apache Storm come in.

3.3 Options for data ingestion

Here are some data ingestion options:

- ▶ Import data into HDP by using Apache Sqoop, which is a tool that is used to transfer bulk data between a relational database server and HDFS. Apache Sqoop supports Kerberos security, compression, parallel import and export, incremental load, full load, and data load directly to Hive. Relational database ingestion can be done by using Apache NiFi too. However, with Sqoop you can specify a table, and in NiFi you must specify a query.
- ▶ Ingest files into a landing server and use the HDFS command-line interface (CLI) to ingest data.
- ▶ Real-time streaming data by using Apache Kafka, Storm, and NiFi. In our proof of concept (PoC), we use Apache NiFi Version 1.5.0.3.1.2.0-7. The interface is similar to Figure 3-5.

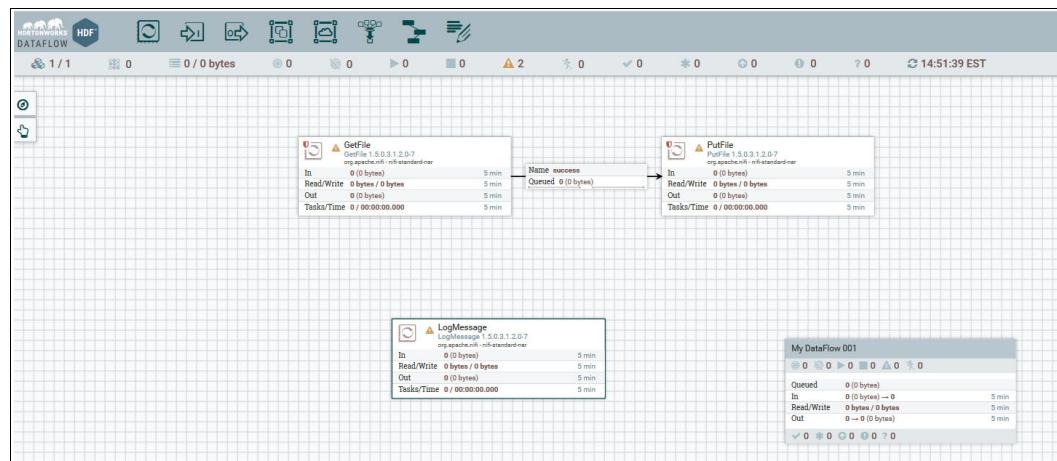


Figure 3-5 Apache NiFi

Apache NiFi is a robust, secure, and performant data delivery and data flow management platform. It helps you access the data that you need from almost any source. It can help you perform event process transformation routing, filtering, prioritization, and securely move that data to its required destination while keeping a traceable and auditable record.

NiFi supports different types of protocols, such as messaging, block, TCP, UDP, and many others. NiFi can run in parallel with other applications, but it performs best when the entire system (or multiple systems in a cluster) is dedicated to it. When collecting real data streams from IoT devices, Apache MiNiFi is used. Apache MiNiFi provides all the key functions of NiFi in varying, smaller footprints in multiple form factors:

- A MiNiFi Java agent
- A C++ agent
- Libraries for Android and iOS

With Apache MiNiFi agents, you can capture intelligence at the edge nodes by performing lightweight operations such as data filtering. MiNiFi is a subproject of NiFi, and NiFi is in the data center. MiNiFi runs on hardware that is probably dedicated to a different primary purpose. Whether this hardware is IoT, a cash register or point of sale system, a car modem, or physical sensors, its job is to process and extract this data while not taking unnecessary resources from the primary function.

- ▶ A combination of NiFi and Kafka. You can use NiFi to read and write from external systems and build quickly a scalable data ingestion pipeline by using a codeless approach. Apache Kafka is used as a low latency scalable publish and subscribe system to decouple data producers and consumers and enable data retention and caching in a streaming architecture.
- ▶ You can also use IBM Watson Machine Learning Accelerator with Spark data frames to read data directly from the datalake by using the IBM Watson Machine Learning Accelerator console.

Figure 3-6 shows how to create a data set by using IBM Spectrum Conductor Deep Learning Impact V1.2.1. IBM Spectrum Conductor Deep Learning Impact supports Lightning Memory-Mapped Database (LMDB), TFRecord, and other data sets.

The screenshot shows the 'IBM Spectrum Computing Cluster Management Console' interface. The top navigation bar includes 'Workload', 'Resources', 'System & Services', and 'Reports & Logs'. The left sidebar has sections for 'Spark', 'Application Instances', 'Notifications', 'Datasets' (which is currently selected and highlighted in blue), and 'Projects'. Under 'Datasets', there are links for 'All datasets', 'New', and 'Delete'. A sub-menu for 'Deep Learning' is open, listing 'Spark Instance Groups', 'My Applications & Notebooks', 'Notebook Management', 'Version Management', 'Anaconda Management', and 'Deep Learning'. The main content area displays a table with columns: Name, Format, State, Size, Spark Instance Group, and Created. A message at the bottom of the table says 'No dataset found'. Below the table, it states 'Showing 0 to 0 of 0 entries'.

Figure 3-6 Deep learning

Depending on the deep learning (DL) framework that you are using, different IBM Spectrum Conductor Deep Learning Impact data set types can be used. If you are using Caffe, you can create the LMDB and images for object classification data sets. If you are using TensorFlow, you can create the TensorFlow Records, Images for object classification, Images for object detection, Images for vector output, CSV files, and other generic data set types.

Figure 3-7 shows the available data set formats in IBM Spectrum Conductor Deep Impact Learning.

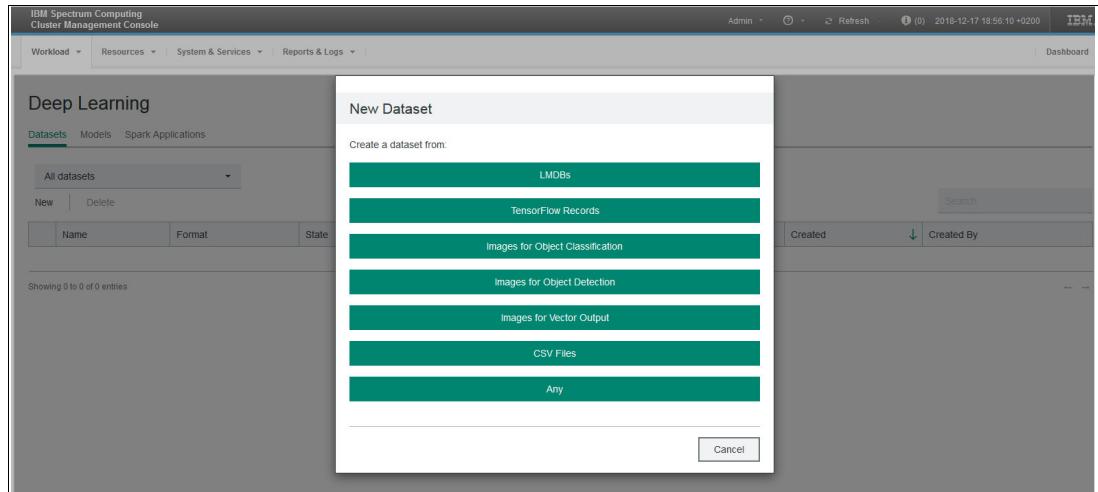


Figure 3-7 All data sets

Figure 3-8 shows an IBM Spectrum Conductor Spark Instance Groups (SIG) that is built with the IBM Spectrum Conductor Deep Learning Impact template.

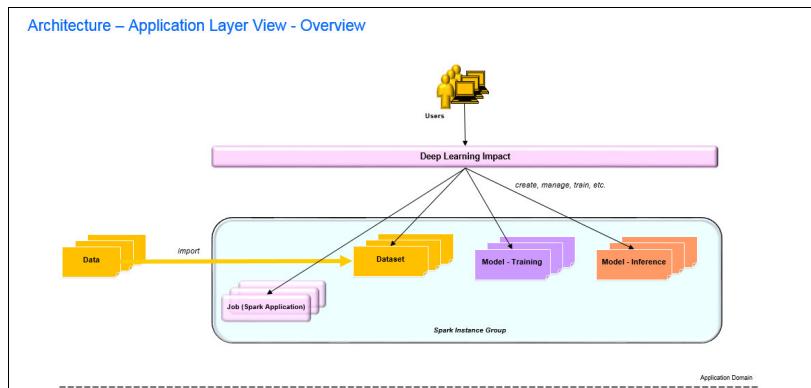


Figure 3-8 Application layer view

3.4 Using data connectors to work with external data sources

Spark is one of the most popular open source tools for data processing in the advanced analytics environment. Before the release of IBM Spectrum Conductor with the Spark feature, if you wanted the hosts in your SIG to connect to an external data source such as IBM Cloud Object Storage (IBM COS), IBM Spectrum Scale, or HDFS, there was a large amount of configuration that was required on those hosts. In the context of IBM Spectrum Conductor, a data connector contains the type, the URI, the authentication method, and all of the required libraries to access the data source.

Data connectors simplify data source administration and configuration, and separate credential management and data usage from applications. Users can optionally make some selections about which data connectors are active and which one to use as the default. For batch applications, these selections are used as the default, but can be changed at run time. For notebook applications, these values are always used.

Figure 3-9 shows that type of data connectors that you can use to connect to IBM Watson Machine Learning Accelerator.

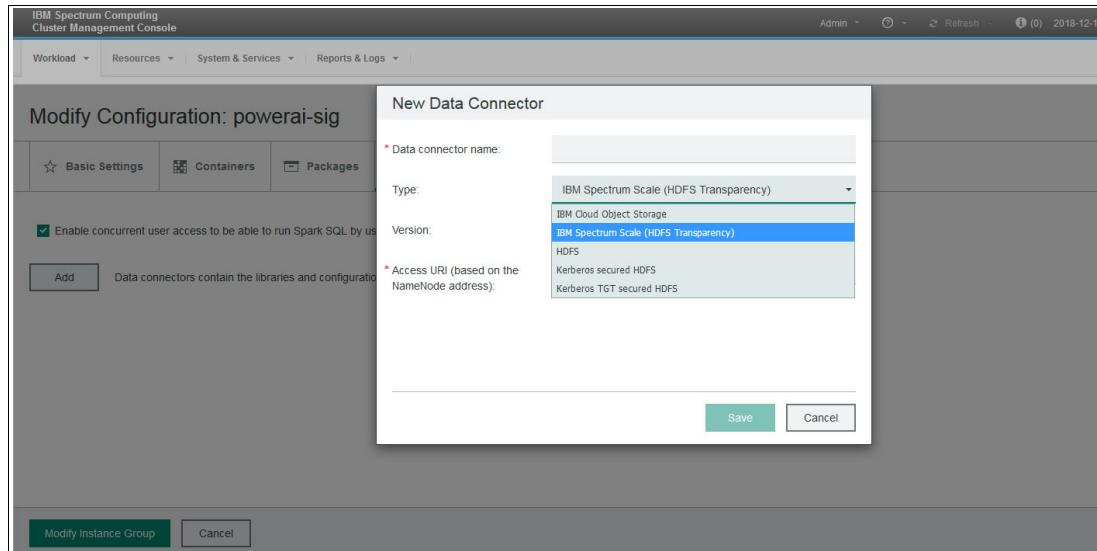


Figure 3-9 Types of data connectors

3.5 How integration improves the artificial intelligence models

The capabilities of POWER9 processors and NVIDIA GPUs in POWER9 processor-based servers can speed data ingestion by using machine intelligence. IBM Power Architecture® is the only NVLink CPU-to-GPU-enabled architecture. With the integration of HDP and HDF together with IBM Watson Machine Learning Accelerator, you can train a model more easily and efficiently. The construction of the MLDL model requires a large amount of historical data to detect a pattern that is related to what you are trying to predict. The best tool to do this task in HDP is Spark. After this task is complete, you can start streaming data-in-motion through NiFi. NiFi sends that data to Storm or Spark Streaming, which applies the model and makes the prediction.



Integration details

This chapter provides more details about the integration steps that were introduced in Chapter 2, “Integration overview” on page 29.

The following topics are covered in this chapter:

- ▶ Integrating IBM Watson Machine Learning Accelerator with Hortonworks
- ▶ Integrating IBM Watson Studio Local, IBM Watson Machine Learning Accelerator, and Hadoop clusters
- ▶ Integrating IBM Watson Studio Local with Hortonworks Data Platform

4.1 Integrating IBM Watson Machine Learning Accelerator with Hortonworks

IBM Watson Machine Learning Accelerator integrates with Hortonworks in two ways:

- ▶ A remote Spark job can be submitted from a Jupyter Notebook running on an IBM Spectrum Conductor Spark Instance Group (SIG) to a Hortonworks cluster that has an Apache Livy service running on it. Sparkmagic is used in the notebook to make the connection to Livy.
- ▶ Data is read by Spark jobs running on IBM Watson Machine Learning Accelerator by using a Hadoop Distributed File System (HDFS) URL directly or by defining an HDFS connector in IBM Spectrum Conductor.

Apache Livy is a service that enables interaction with a Spark cluster over a REST interface. It supports long-running Spark sessions and multi-tendency. For more information, see [Apache Livy - A REST Service for Apache Spark](#).

Sparkmagic runs in a Jupyter Notebook. It is a set of tools for interactively working with remote Spark clusters through Apache Livy. For more information, see [GitHub - Jupyter magics and kernels for working with remote Spark clusters](#).

One of the major building blocks for running an application on IBM Watson Machine Learning Accelerator is a SIG. Before you create a SIG, you must complete a set of prerequisite steps. For more information about SIGs, see [IBM Knowledge Center](#).

The first step in building an IBM Watson Machine Learning Accelerator cluster is defining a *resource group* (RG), which contains a set of resources that is part of the IBM Watson Machine Learning Accelerator cluster. RGs provide a simple way of organizing and grouping resources (hosts). Instead of creating policies for individual resources, you create and apply them to an entire group. A cluster administrator can define multiple RGs, assign them to consumers, and configure a distinct resource plan for each group.

RG host lists are defined by a static list or are dynamic. A static list is where you specifically choose which hosts to use. A dynamic list is where you specify either all hosts or a resource requirement so that when new hosts join they are automatically added to the RG.

If you plan to run a GPU-dependent application and have a GPU-based host in the cluster, make them part of the RG too. In this case, create two RGs: one with CPU cores and the other with GPUs.

For the RG with the GPU hosts, ensure that all the hosts in the RG have a number value in the *ngpus* column and that the number in the *Total slots for this group* column is equal to the total number of GPUs on all the hosts in the RG. RGs can be static or dynamic by using different host attributes to define membership or by using tags. Resources can be logical entities that are independent of nodes (bandwidth capacity and software licenses). Those resources are used by consumers.

A *consumer* is a logical structure that creates the association between the workload demand and the resource supply. Consumers are organized hierarchically into a tree structure to reflect the structure of a business unit, department, projects, and so on. You create a consumer and assign a set of RGs to it.

Consumers also help set up different roles for different users to use the resources as needed, and have demarcated differences between them. For example, two departments in an organization (sales and finance) can run different Spark applications for their individual needs with different sets of resources. There can also be resources that can be shared among the different departments based on their business requirements.

4.1.1 Running remote Spark jobs with Livy

This section shows how to run a job remotely on a Hortonworks Spark cluster by using Apache Livy and Sparkmagic. Figure 4-1 shows an IBM Watson Machine Learning Accelerator Jupyter Notebook that uses Sparkmagic to connect to the Apache Livy service running on an HDP edge node.

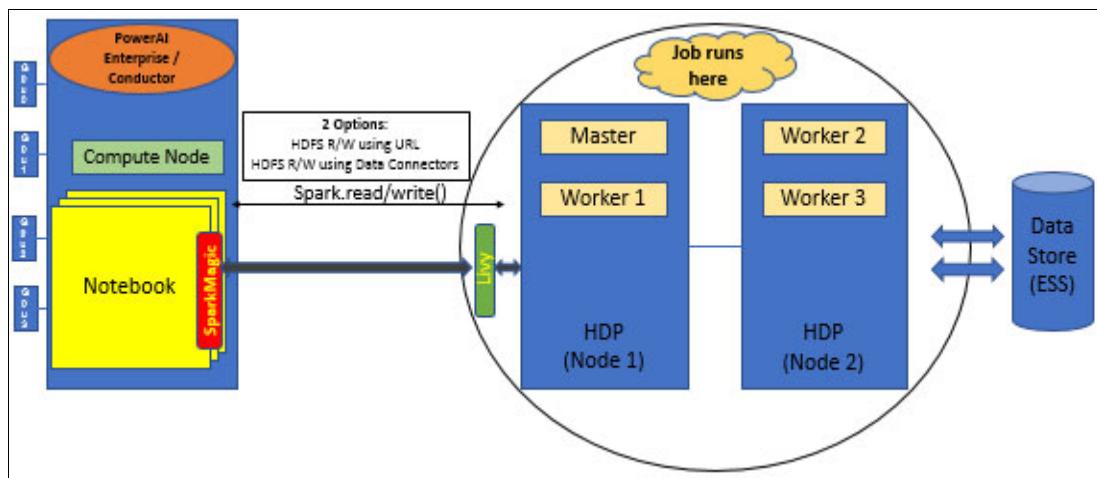


Figure 4-1 IBM Watson Machine Learning Accelerator with Hortonworks and Livy

On the Hadoop side, install Livy2 on an edge node, which is a node that includes the Hadoop clients for Spark, Hive, and HDFS. The Hadoop distribution vendors typically have detailed documentation about how to accomplish this task. For Hortonworks Data Platform (HDP) V2.6.5, we followed the instructions that are found at [Installing Livy - Hortonworks Data Platform](#).

Install a custom Jupyter Notebook with Sparkmagic on IBM Watson Machine Learning Accelerator by following the instructions that are found in Appendix B, “Installing an IBM Watson Machine Learning Accelerator notebook” on page 121.

The following example uses the Spark MLlib binomial logistic regression code sample from [Classification and regression - Spark 2.2.0 Documentation](#). In this case, we use a different data set for the data that is accessed from HDFS.

Logistic regression is a machine learning (ML) method that is used to describe data and explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables.

Criteo is a data set about an ad click-through rate. It contains data about users and the pages they are visiting and calculates the probability that they click certain ads. The data that is used in this example consists of a portion of Criteo's traffic over 7 days. Each row corresponds to a display ad that is served by Criteo. Positive (clicked) and negatives (non-clicked) examples were subsampled at different rates to reduce the data set size. The examples are chronologically ordered.

To install the data into HDFS, run the following commands on one of the nodes in the Hadoop cluster:

- ▶ **wget**
<https://s3-us-west-2.amazonaws.com/criteo-public-svm-data/criteo.kaggle2014.svm.tar.gz>
- ▶ **tar -xzvf criteo.kaggle2014.svm.tar.gz**
- ▶ **hdfs dfs fs -put criteo.kaggle2014.svm <TARGET DIRECTORY>**

In our sample, the target directory is /user/user1/dataset.

Example 4-1 shows the Spark MLlib code running remotely on the Hadoop cluster and reading the data from the HDFS file system. This is an example of data locality, that is, running the application close to where the data is.

Example 4-1 Code snippet to perform logistic regression by using Sparkmagic

```
%load_ext sparkmagic.magics

%spark add -s test -l python -u http://129.40.2.75:8999 -a u -k

%%spark
from pyspark.ml.classification import LogisticRegression

# Load training data from HDFS
training =
spark.read.format("libsvm").load("/user/user1/datasets/criteo.kaggle2014.test.svm")
)

%%spark

# Create a logistic regression model
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model training data using the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for logistic regression
print("Coefficients: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))
```

4.1.2 Accessing the Hadoop data from IBM Watson Machine Learning Accelerator

Figure 4-2 shows a Spark job running on an IBM Watson Machine Learning Accelerator Jupyter Notebook accessing data on a Hadoop cluster. There are a couple options for reading HDFS data in IBM Watson Machine Learning Accelerator: It can be read into a Spark data frame directly by using HDFS URL or by defining an HDFS connector. The latter option includes options for Kerberos.

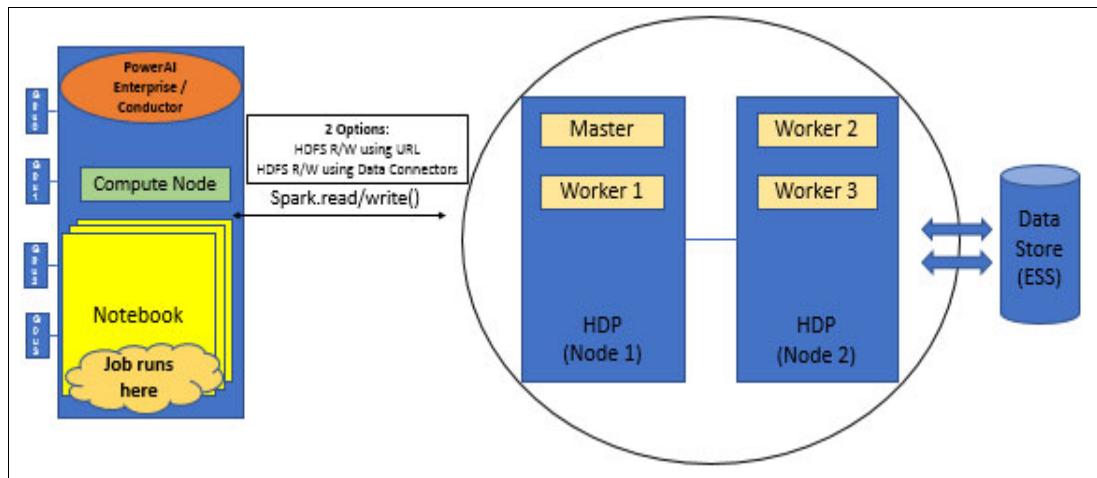


Figure 4-2 Hadoop data access from IBM Watson Machine Learning Accelerator

In the following example, we reuse the logistic regression example with the Criteo data set. The model runs locally on IBM Watson Machine Learning Accelerator with the data coming from the remote Hadoop cluster.

Example 4-2 shows the Spark reading of the data by using the HDFS URL.

Example 4-2 Code snippet to perform logistic regression by using remote data

```
import time
from pyspark.ml.classification import LogisticRegression

# Load training data from the remote Hadoop cluster
training =
spark.read.format("libsvm").load("hdfs://ib-hdprb001.pbm.ihost.com:8020/user/user1
/datasets//criteo.kaggle2014.test.svm")

# note sure if this analysis makes any sense with this type of data
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for logistic regression
print("Coefficients: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))

# We can also use the multinomial family for binary classification
mlr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8,
family="multinomial")
```

```
# Fit the model  
mlrModel = mlr.fit(training)
```

An HDFS data connector can be defined by updating the IBM Watson Machine Learning Accelerator SIG configuration. The SIG must be stopped to make configuration changes. The SIG details include a pane for defining data connectors.

Figure 4-3 shows the data connector pane. Click **Add** to define the data connector.

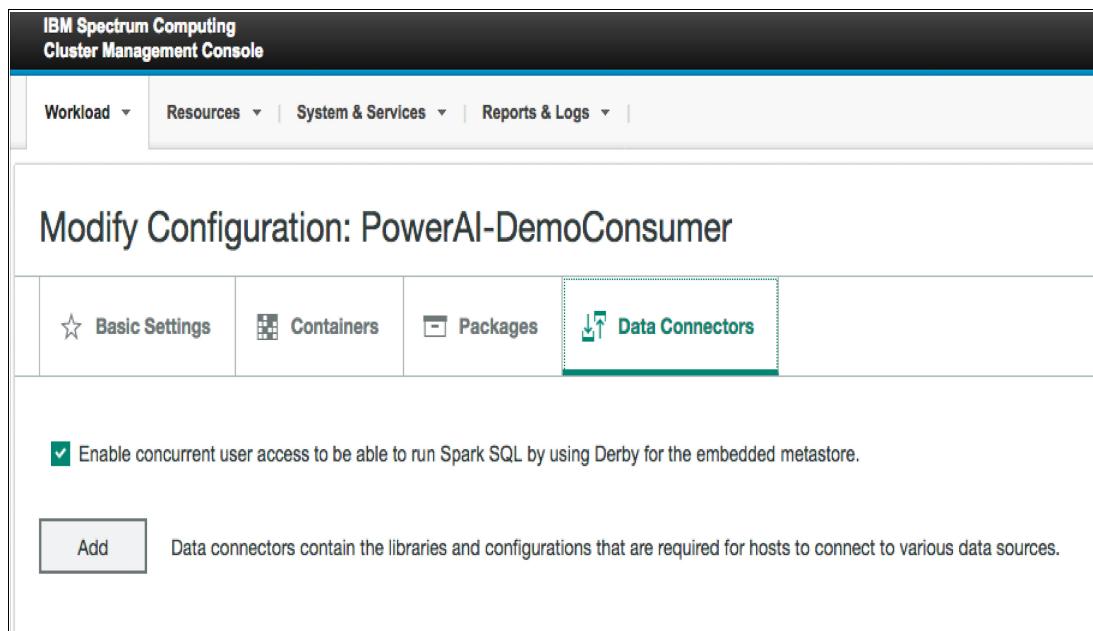


Figure 4-3 Adding a data connector in an IBM Watson Machine Learning Accelerator SIG

Provide a name, type, and access URI for the new data connector. Figure 4-4 shows details for the new data connector.

The screenshot shows a 'New Data Connector' dialog box. At the top, it says 'New Data Connector'. Below that, there are four input fields: 'Data connector name' with the value 'hdp-hdfs', 'Type' set to 'HDFS', 'Version' set to '2.7', and 'Access URI (based on the NameNode address)' with the value 'hdfs://:8020'. At the bottom right are two buttons: a green 'Save' button and a white 'Cancel' button.

Figure 4-4 New Data Connector details

To use the new connector, save it, redeploy it, and start the SIG. The portion of the notebook that reads the remote data into a Spark data frame can now specify the location directly without the HDFS URL, as shown in Example 4-3.

Example 4-3 Loading data by using an HDFS connector

```
# Load training data from the remote Hadoop cluster
training =
spark.read.format("libsvm").load("/user/user1/datasets//criteo.kaggle2014.test.svm")
```

This section covered IBM Watson Machine Learning Accelerator integration with Hadoop. We showed how to run a Spark job on the Hadoop cluster by using Sparkmagic in a notebook and how to read HDFS data into a local Spark job.

4.2 Integrating IBM Watson Studio Local, IBM Watson Machine Learning Accelerator, and Hadoop clusters

This section shows how to integrate IBM Watson Studio Local with IBM Watson Machine Learning Accelerator and Hadoop clusters.

Figure 4-5 illustrates the interaction architecture between IBM Watson Studio Local and IBM Watson Machine Learning Accelerator. Livy is installed as an application on IBM Watson Machine Learning Accelerator and Sparkmagic is used in a IBM Watson Studio Local notebook to create a connection for Spark job execution on IBM Watson Machine Learning Accelerator.

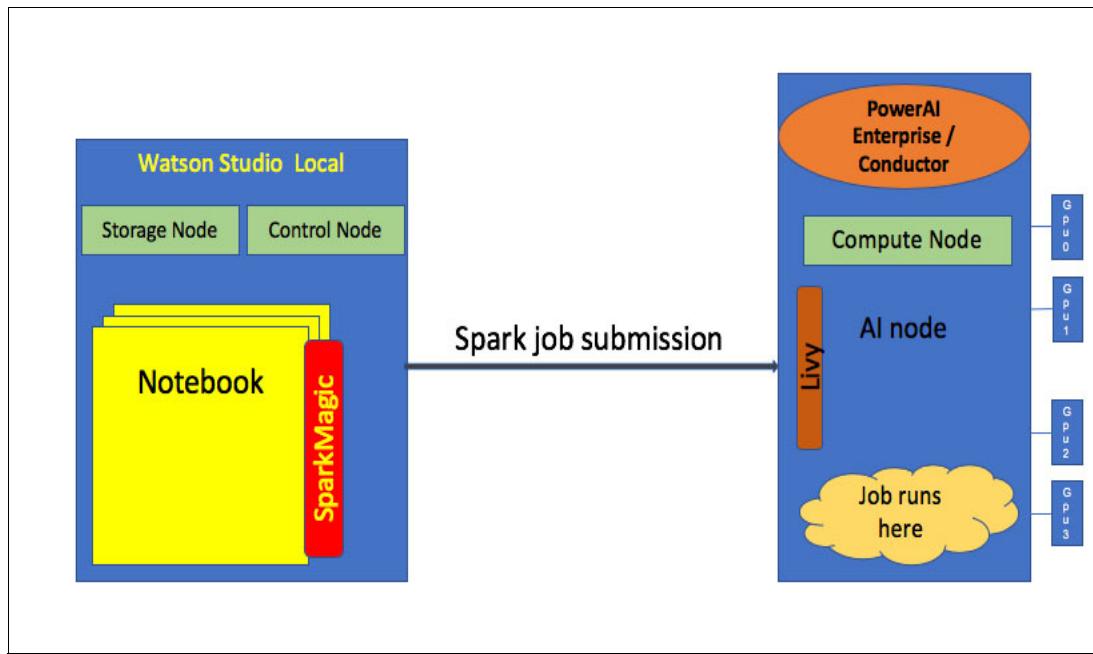


Figure 4-5 IBM Watson Studio Local with IBM Watson Machine Learning Accelerator

To integrate IBM Watson Studio Local notebooks with IBM Watson Machine Learning Accelerator, install a Livy application on the IBM Spectrum Conductor SIG, as described at the [cws-livy GitLab project](#).

Here are the application installation step details:

1. From the IBM Spectrum Computing Cluster Management Console, click **Workload** → **Application Instances**, and then click **New** to register an application, as shown in Figure 4-6.

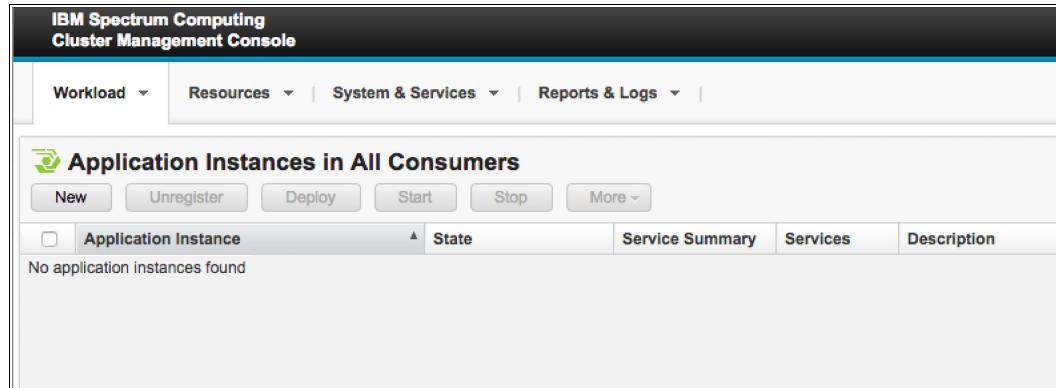


Figure 4-6 Registering a new application instance

2. In the New Application Instance dialog, click **Browse** to select the `Livy_0.5.0.yaml` application template file that was downloaded from the cws-livy GitLab project, as shown in Figure 4-7.

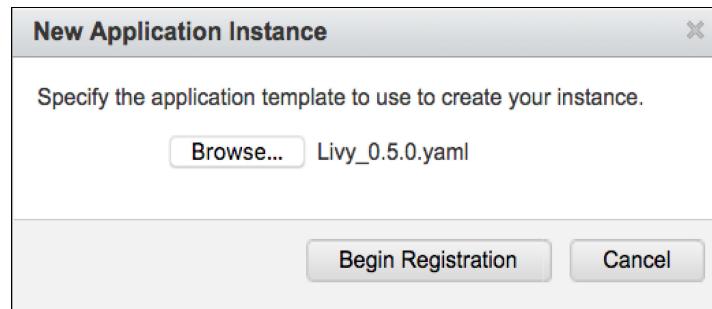


Figure 4-7 Specifying the application template

3. Next, you are guided through the New Application wizard:
 - a. For the application instance name, specify Livy2, as shown in Figure 4-8.

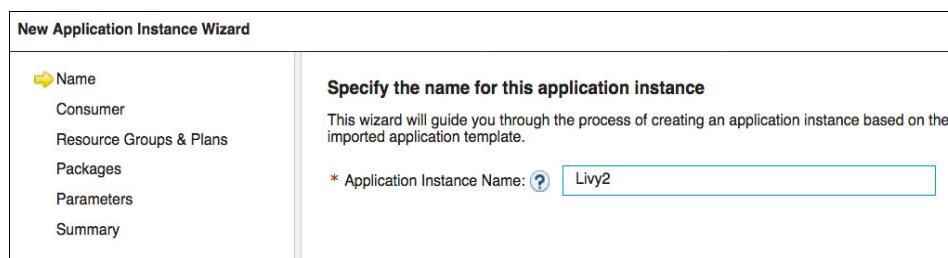


Figure 4-8 Specifying an application name

- b. For Consumer, select the one that is associated with the SIG. In this case, we had a SIG called Notebook and the consumer has the same name, as shown in Figure 4-9.

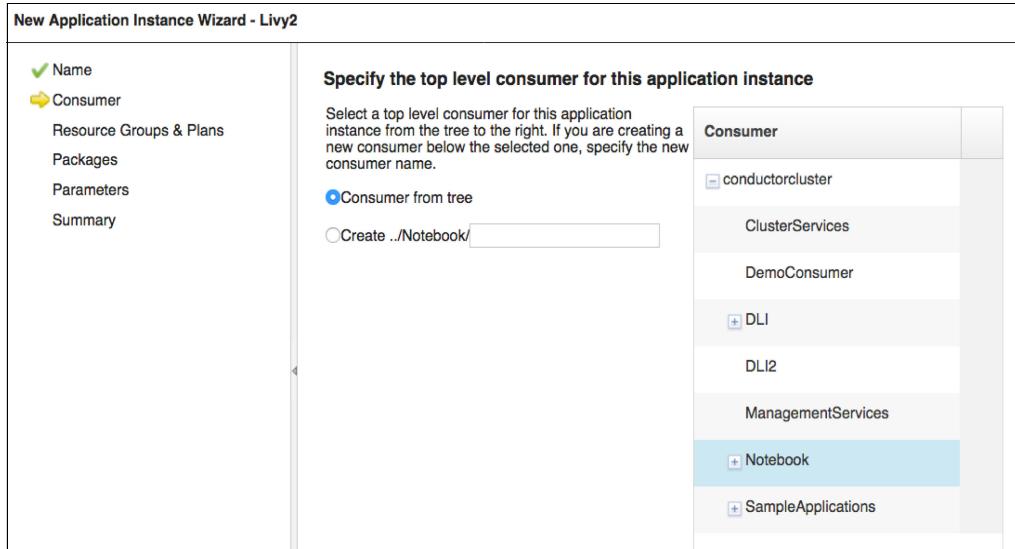


Figure 4-9 Specifying the top-level consumer for the application

- c. For Resource Group and Plans, select the resource that you used for the Spark master in the SIG definition. The resources are usually the computer hosts, as shown in Figure 4-10. The Resource Groups and Plans selection box might not appear until you hover the cursor over the location.

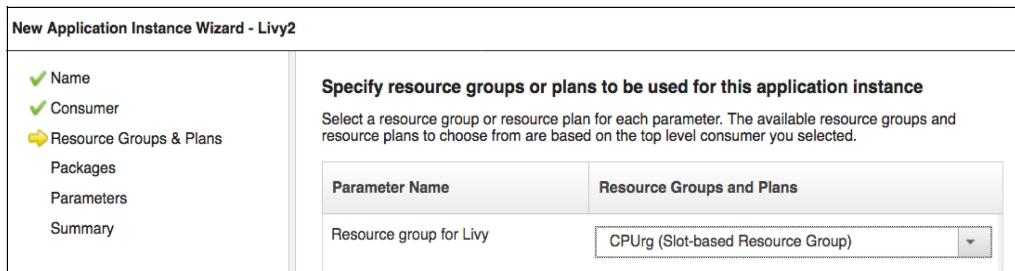


Figure 4-10 Selecting the resource groups

- d. For the repository package, specify `livy_0_5_0_package.tar.gz`, which was downloaded from the cws-livy GitLab project, as shown in Figure 4-11.

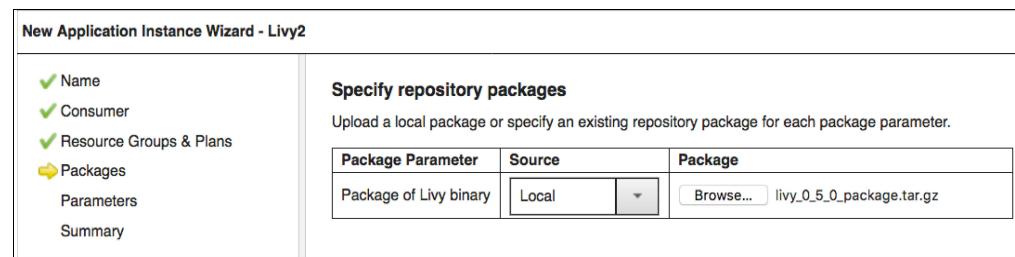


Figure 4-11 Specifying the repository packages

- e. Specify the parameter values. For the port number, 8998 is a typical default. If you install the Livy application on multiple SIGs, you need a unique port number for each SIG.

The deployment directory should be unique. We used the deployment directory of the SIG and added /livy2 to it.

The execution user should be the same as the one that you selected for your SIG. If the SIG was built by using the IBM Spectrum Conductor Deep Learning Impact template, the execution user is the ego admin user, typically called egoadmin. For our test, we used an OS user who is called demouser to run the SIG.

The **SPARK_HOME** and Spark Master URL details can be found in the associated SIG. For our test, we had a Notebook SIG defined. We went to the list of SIG list by clicking **Workload → Spark → Spark Instance Groups**. Then, we selected the Notebook SIG to see the details for the Spark Deployment and Master URL. For an example the parameters, see Figure 4-12.

Parameter Name	Value
Port number	8998
Deployment directory	/cwsshare/demouser/notebook/spark/livy2
Execution user	demouser
SPARK_HOME	/cwsshare/demouser/notebook/spark/spark-2.3.1-hadoop
URL to the Spark Master starting from spark://...	

Figure 4-12 Specifying the parameter values associated with the SIG

- f. The last window of the wizard provides a summary of the inputs. Click **Register**. It can take several minutes for the application registration to complete.
- g. Click **Done** to exit the wizard after the application instance successfully registers.

The application now is in a Registered state, as shown in Figure 4-13.

New	Unregister	Deploy	Start	Stop	More
Application Instance	State	Service Summary	Services	Description	
Livy2	Registered	1	1	Application template for Livy V0.5.0	

Figure 4-13 The Livy2 application in the Registered state

- h. Select the check box in the Livy2 application row and click **Deploy**. You do not need to provide a timeout value in the deploy question message box, so click **Deploy** again.

- i. After deployment completes, select the check box in the Livy application row and click **Start**. Click **Start** again in the start question box.
- j. Once started, the application has a green Service Summary and shows the URL for the Livy2 application, as shown in Figure 4-14. The URL is what you need to specify in the IBM Watson Studio Local notebook that connects to Livy by Sparkmagic.

Application Instance	State	Service Summary	Services	Description
Livy2	Started	1	1	Application template for Livy V0.5.0

Application Instance: Livy2

General

Application Instance: Livy2	State: Started
Service Summary:	Description: Application template for Livy V0.5.0
Registered Time: 01/26/2019, 14:50:15 MST	Last Deploy Time: 01/26/2019, 14:55:43 MST
Start Time: 01/26/2019, 14:59:00 MST	Last Stop Time: -
Up Time: 00:00:04	Top Consumer: /Notebook
UUID: a91939c4-4f80-497b-b8ad-36fa304cc380	New Consumers: /Notebook/Livy2-livyserver

Outputs

livy_URL:

Figure 4-14 The started Livy2 Application showing the URL in the outputs

On IBM Watson Studio Local, create a notebook to test the connection to the Livy service by completing the following steps:

1. In the first notebook cell, load the Sparkmagic extension by running the following command:

```
%load_ext sparkmagic.magics
```
2. In the second notebook cell, create a Livy session to the IBM Watson Machine Learning Accelerator SIG by running the following command. Use the URL that you obtained while creating the Livy application instance.

```
%spark add -s <session_name> -l python -u <livy_URL> -a u -k
```
3. In the third notebook cell, run Spark code in the notebook by running the magic `%%spark -l <language>` command, where the language for the Livy session is either Python, Scala, or R. The default language is Python. In our example, we used the Spark Context (`sc`) to create a distributed collection of a 1000 elements and then counted the elements (`ans: 1000`).

```
%%spark
sc.parallelize(range(1000)).count()
```
4. After the application run completes, clean up the Livy session to release the associated resource by running the following command:

```
%spark cleanup
```

The ran test is show in Figure 4-15.

```
In [1]: %load_ext sparkmagic.magics
In [2]: %spark add -s notebook -l python -u http://:8998 -a u -k
Starting Spark application
ID YARN Application ID Kind State Spark UI Driver log Current session?
0 None pyspark idle ✓
SparkSession available as 'spark'.
In [3]: %%spark
sc.parallelize(range(10000)).count()
10000
In [4]: %spark cleanup
```

Figure 4-15 The IBM Watson Studio Local to IBM Watson Machine Learning Accelerator Livy Notebook Test

After the Livy connection is verified, more sophisticated deep learning (DL) models can be run. Example 4-4 shows the TensorFlow Keras Fashion MNIST model running on IBM Watson Machine Learning Accelerator from an IBM Watson Studio Local notebook. The Fashion MNIST example is described in the [TensorFlow basic classification documentation](#).

The In [#]: prefixes represent the Jupyter Notebook cell numbers in which the code is run.

Example 4-4 Running the TensorFlow Keras model

```
In [1]: %load_ext sparkmagic.magics
In [2]: %spark add -s test -l python -u http://icpc2.rch.stglabs.ibm.com:8998 -a u -k
In [3]: %%spark
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
# Helper libraries
import matplotlib.pyplot as plt
print(tf.__version__)
In [4]: %%spark
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
In [5]: %%spark
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
print("train images shape: ", train_images.shape)
print("train labels count: ", len(train_labels))
In [6]: %%spark
print("test images shape: ", test_images.shape)
print("test labels count: ", len(test_labels))
In [7]: %%spark
train_images = train_images / 255.0
test_images = test_images / 255.0
```

```

In [8]: %%spark
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
In [9]: %%spark
predictions = model.predict(test_images)
predictions[0]
In [10]: %spark cleanup
In [11]: #@title MIT License
#
# Copyright (c) 2017 François Chollet
#
# Permission is hereby granted, free of charge, to any person obtaining a
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.
tf_fashion_minst_paie_livy.jupyter
http://localhost:8888/nbconvert/html/redbook-watson-studio-local/tf...
2

```

4.3 Integrating IBM Watson Studio Local with Hortonworks Data Platform

Figure 4-16 illustrates a typical interaction architecture between IBM Watson Studio Local and HDP.

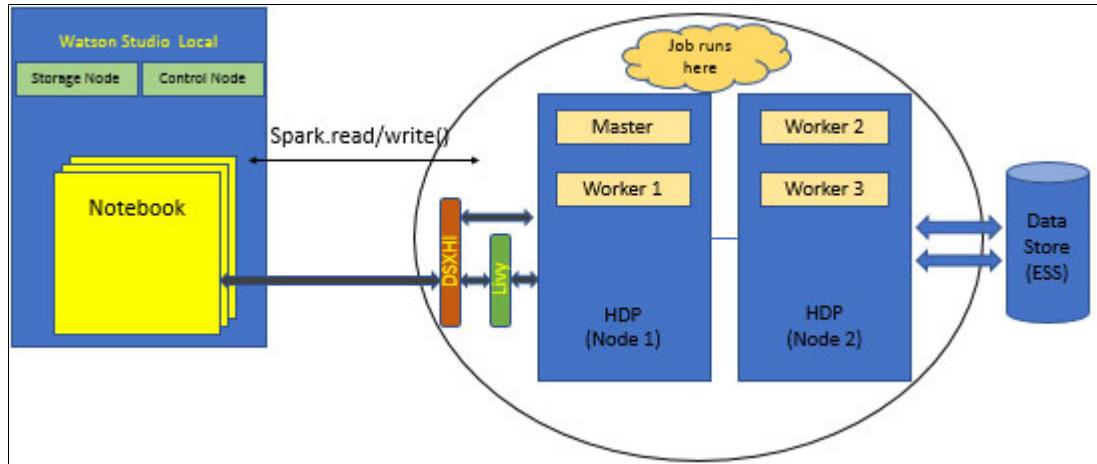


Figure 4-16 IBM Watson Studio Local with Hortonworks

IBM Watson Studio Local provides a Hadoop registration service that eases the setup of an HDP cluster for IBM Watson Studio Local. Using Hadoop registration is the preferred approach because it provides more functions for scheduling jobs as YARN applications.

To integrate IBM Watson Studio Local with HDP, complete the following steps:

1. Install the DSXHI (Hadoop registration gateway service and Hadoop registration rest service) in one of the edge nodes of the HDP by using the steps that are detailed in [IBM Knowledge Center](#).
2. On the IBM Watson Studio Local browser, click **Admin Console → Hadoop Integration**.

3. Add the HDP node by using the **Add Registration** window, which is shown in Figure 4-17.

The screenshot shows a 'Add Registration' dialog box. It has three text input fields: 'Display Name *' containing 'pw8002', 'Service URL *' containing 'https://ib-hdprb001.pbm.ihost.com:8443/gateway/p460a10', and 'Service User ID *' containing 'dsxhi'. At the bottom right are two buttons: 'Cancel' and 'Add'.

Figure 4-17 Add Registration window

Adding the registration integrates IBM Watson Studio Local with the HDP nodes for remote Spark job submission.

4. After registration is complete, you can view details about the registered Hadoop cluster, push runtime images to the Hadoop cluster, and work with images. For more information, see [Hadoop Integration](#).
5. After the runtime environment push completes, use a IBM Watson Studio Local notebook to work with the Hadoop cluster. Example 4-5 shows the import of the `dsx_core_utils` package and the call to `dsx_core_utils.get_dsxhi_info()`.

Example 4-5 Notebook code to get the list of Hadoop clusters

```
import dsx_core_utils  
DSXHI_SYSTEMS = dsx_core_utils.get_dsxhi_info(showSummary=True)
```

The result of the `get_dsxhi_info` call shows a list of the Hadoop clusters, type of service (livyspark2), and the image ID names. Example 4-6 shows how the result appeared on our test system.

Example 4-6 List of available Hadoop systems

```
Available Hadoop systems:  
systemName LIVYSPARK LIVYSPARK2 imageId  
0 pw8002 livyspark2 dsx-scripted-ml-python2  
1 pw8002 livyspark2 dsx-scripted-ml-python3  
2 pw8002 livyspark2 py-tqdm-1.0-dsx-scripted-ml-python2  
3 pw8002 livyspark2 tensorflow-1.12-dsx-scripted-ml-python3  
4 pw8002 livyspark2 tf-numpy-1.12-dsx-scripted-ml-python3
```

6. You can configure the Spark session that is when running on the selected registered Hadoop Integration system. Example 4-7 shows an example configuration.

Example 4-7 Configuring a Spark session

```
myConfig={  
    "queue": "default",  
    "driverMemory": "2G",  
    "numExecutors": 2  
}
```

7. Run **dsx_core_utils.setup_livy_sparkmagic** to set up Sparkmagic to connect to the selected Hadoop Integration System, as shown in Example 4-8.

Example 4-8 Setting up Sparkmagic

```
# Set up sparkmagic to connect to the selected registered HI  
# system with the specified configs. **NOTE** This notebook  
# requires Spark 2, so you should set 'livy' to 'livyspark2'.
```

```
dsx_core_utils.setup_livy_sparkmagic(  
    system="pw8002",  
    livy="livyspark2",  
    imageId="dsx-scripted-ml-python2",  
    addlConfig=myConfig)  
# (Re-)load spark magic to apply the new configs.  
%reload_ext sparkmagic.magics
```

8. Create a remote Livy session to connect to that Hadoop Integration System by using the following command:

```
%spark add -s <session> -l python -u <livy_URL> -a u -k
```

Example 4-9 shows the command to connect to our test HDP cluster.

Example 4-9 Starting the remote Livy session

```
session_name = 'tfsess1'  
livy_endpoint =  
'https://ib-hdprb001.pbm.ihost.com:8443/gateway/p460a10/livy2/v1'  
webhdfs_endpoint =  
'https://ib-hdprb001.pbm.ihost.com:8443/gateway/p460a10/webhdfs/v1'  
%spark add -s $session_name -l python -k -u $livy_endpoint
```

A successful start shows a table that includes the Livy session ID and the YARN application ID for **pyspark**.

Any cell in the notebook that has **%spark** as its first line runs the job remotely on the Hadoop system.

You can access any data from the HDFS file system by using functions like **spark.read.format()** to read the data frames from the file system.

Example 4-10 shows a Spark MLlib sample run.

Example 4-10 Running Spark on the Hadoop cluster from the IBM Watson Studio Local notebook

```
%%spark  
from pyspark.ml.classification import LogisticRegression  
  
# Load training data
```

```

training =
spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")

lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for logistic regression
print("Coefficients: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))

```

9. Clean up the remote Livy session by running the following command, which terminates the session and releases resources back to the remote Hadoop Integration system:

```
%spark cleanup
```

Image customization

IBM Watson Studio Local uses containers to manage the complete lifecycle of the images of the applications, which provide the complete benefits of containers to the IBM Watson Studio Local environment.

IBM Watson Studio Local supports creating custom images and pushing those images to systems that are registered through Hadoop Integration.

To customize an environment by using the Python tqdm library complete the following steps. This process can be extended to other packages.

1. Start the environment.

From your project home page, use the **Environments** tab to start a Jupyter with Python 2.7 environment if it is not already running (a green dot indicates that the environment is running).

The screenshot shows the IBM Data Science Experience Local interface. At the top, there's a navigation bar with icons for Home, Projects, and a dropdown for the current project 'hdp_integration'. Below the navigation bar, there are tabs for Assets (26), Data Sources (3), Jobs (0), Environments (7), and Collaborators (3). The 'Environments' tab is currently selected. On the left, there's a sidebar with a 'Environments' section containing 'Running' (4) and 'Stopped' (3) environments. To the right, there's a search bar labeled 'Search by environment name'. A tooltip is displayed over one of the environment cards, which is titled 'Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2'. The tooltip also shows resource details: CPU CORES (1), GPU (1), GB MEMORY (16GB), and a DATE STARTED of 27 Nov 2018, 8:20 AM. At the bottom of the tooltip are several small icons for managing the environment.

Figure 4-18 Starting the environment

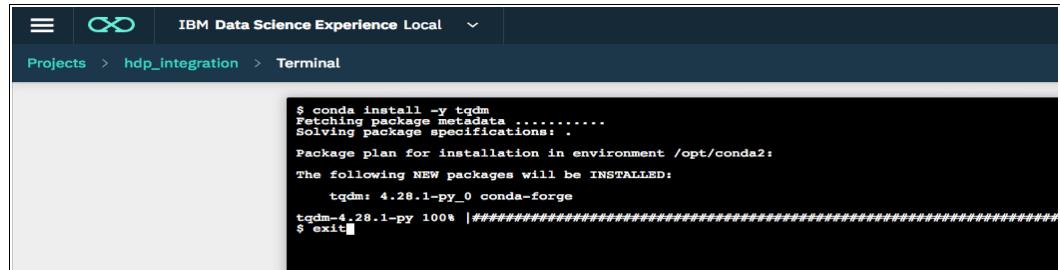
2. Install the package by using the environment's terminal.

From your project home page, use the **Environments** tab to start a terminal shell for the environment that you started in step 1 on page 82.

When you are inside the terminal, run the following command to install tqdm:

```
conda install tqdm -y
```

When the command completes, you can exit the terminal, as shown in Figure 4-19.

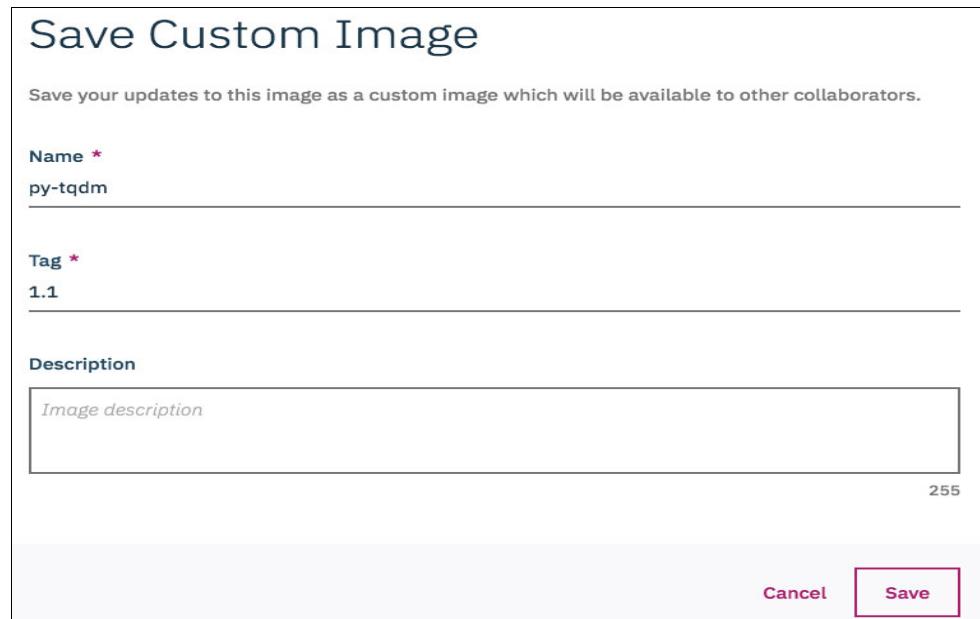


```
$ conda install -y tqdm
Fetching package metadata .....
Solving package specifications: .
Package plan for installation in environment /opt/conda2:
The following NEW packages will be INSTALLED:
    tqdm: 4.28.1-py_0 conda-forge
tqdm-4.28.1-py 100% |#####
$ exit
```

Figure 4-19 Installing the package

3. Save the environment as a custom image.

From your project home page, use the **Environments** tab to save the environment that you edited in step 2 on page 82.



Save Custom Image

Save your updates to this image as a custom image which will be available to other collaborators.

Name *

Tag *

Description

255

Cancel Save

Figure 4-20 Saving the image

4. Select the **Hadoop Integration** option from the Admin Console, as shown in Figure 4-21.

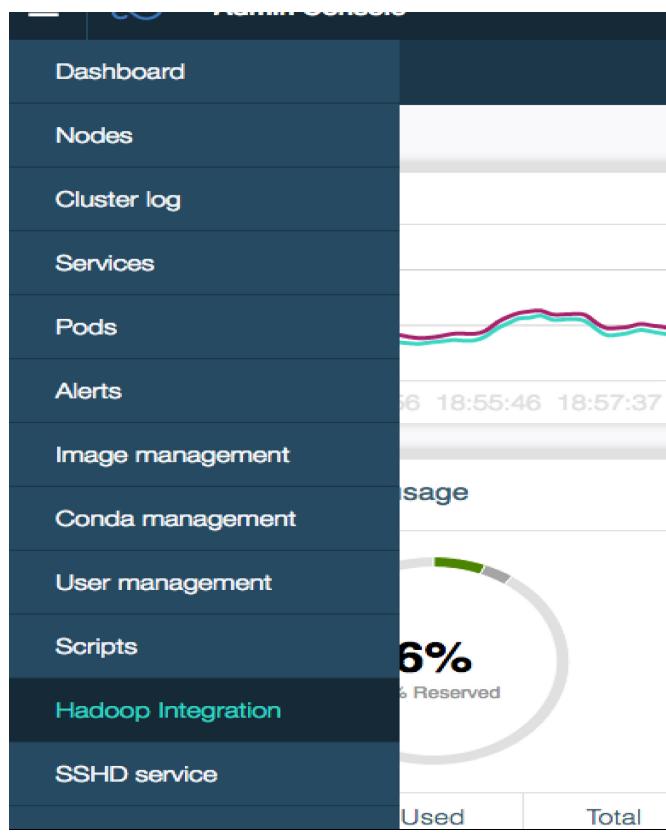


Figure 4-21 Selecting Hadoop Integration from the admin console

5. Display the details, as shown in Figure 4-22.

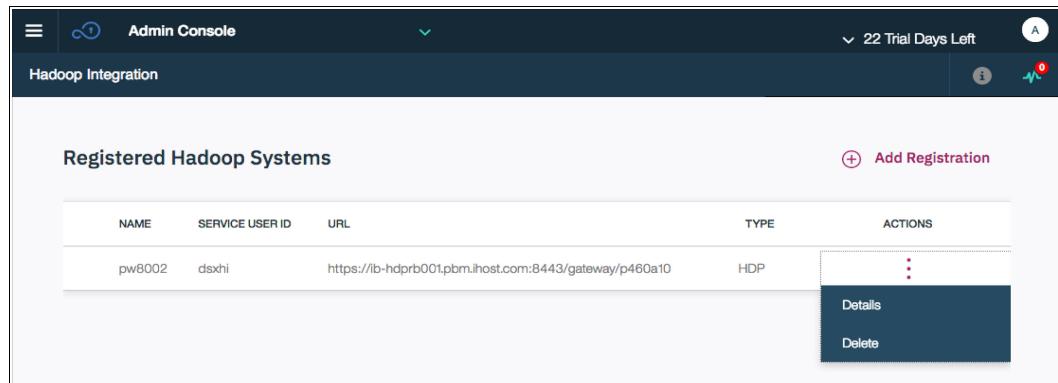


Figure 4-22 Displaying the details of the registered Hadoop system

- Push the run time to the Hadoop cluster, as shown in Figure 4-23.

NAME	TYPE	STATUS	LOG	ACTIONS
py-tqdm-1.1-dsx-scripted-mil-python2	Python 2.7	Not pushed		Push

Figure 4-23 Pushing the run time

- Select the new image in the notebook.

The Hadoop Integration utilities provide a list of images that can be used when connecting to the Hadoop cluster. Select the image to use when connecting to the cluster.

Working with Hadoop data

IBM Watson Studio Local supports the addition of many data sources and remote data sets, including Hive and HDFS for Hadoop clusters. For more information, see [Add data sources and remote data sets](#).

You can shape and refine the data sets by using the [Data Refinery](#).



Accessing real-time data

This chapter provides information about tools that fetch and work with data in real time.

Not all big data analysis is done by using historical data that is stored in a database. Some analytics and machine learning (ML) jobs are performed by using real-time data while they still are “hot” data. This data comes from various sources such as Internet of Things (IoT); and transactional, stock market, and social media data in various different formats, such as JSON, plain text, databases, CSV files, PDF files, images, videos, and audio files.

The Apache Hadoop project has many tools that collect and manage real-time or streaming data.

The following topics are covered in this chapter:

- ▶ Hortonworks DataFlow
- ▶ Apache NiFi
- ▶ Apache Storm
- ▶ Apache Spark Streaming
- ▶ Apache Kafka Streams
- ▶ Integrating streaming tools with data science tools

5.1 Hortonworks DataFlow

Hortonworks DataFlow (HDF) is a scalable platform for real-time streaming analytics that ingests, processes, and analyzes data from various sources to obtain immediately information and valuable insights. HDF addresses key challenges that enterprises face regarding real-time streaming processing (data-in-motion) with high volume and high scale, data provenance and data tracking from IoT devices, edge applications, and various models of streaming sources.

The biggest challenge for enterprises that work with streaming data is to obtain data quickly and securely while clearly tracing the data. HDF provides these features through Apache NiFi and MiNiFi, which address these challenges by providing visibility into real-time operation, control, and data flow management.

HDF is a versatile tool that you can use to build data-in-motion apps and put them into production faster.

With the data-at-rest solution that is provided by Hortonworks Data Platform (HDP), HDF provides you with all the tools that you need to create an end-to-end solution for real-time data analytics. With both HDP and HDF, you have the necessary tools to acquire and ingest data in a fast and secure way, the distributed storage and distributed databases to store this massive data, and the tools to filter and clean the data that removes all the dirty and unnecessary data. There are tools to do ML tasks on the data and find patterns between the data. Each step is secured by single sign-on and audited, and there is a way to track the entire application data flow.

Figure 5-1 gives an overview of the HDP components.

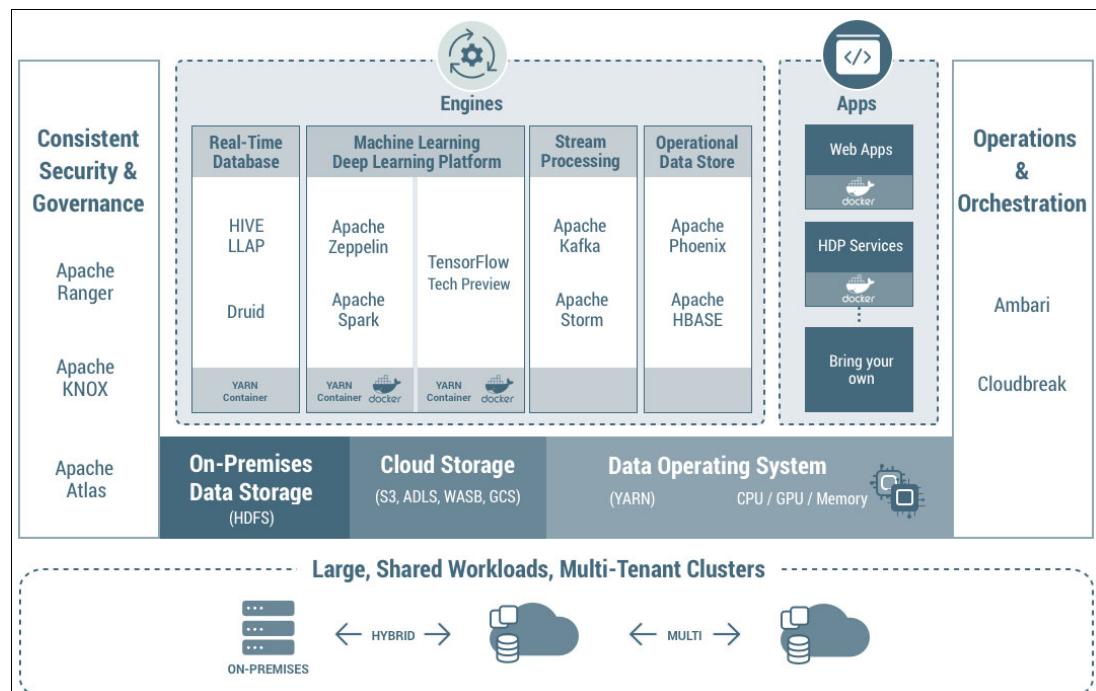


Figure 5-1 Hortonworks Data Platform components

Figure 5-2 gives an overview of the HDF components.

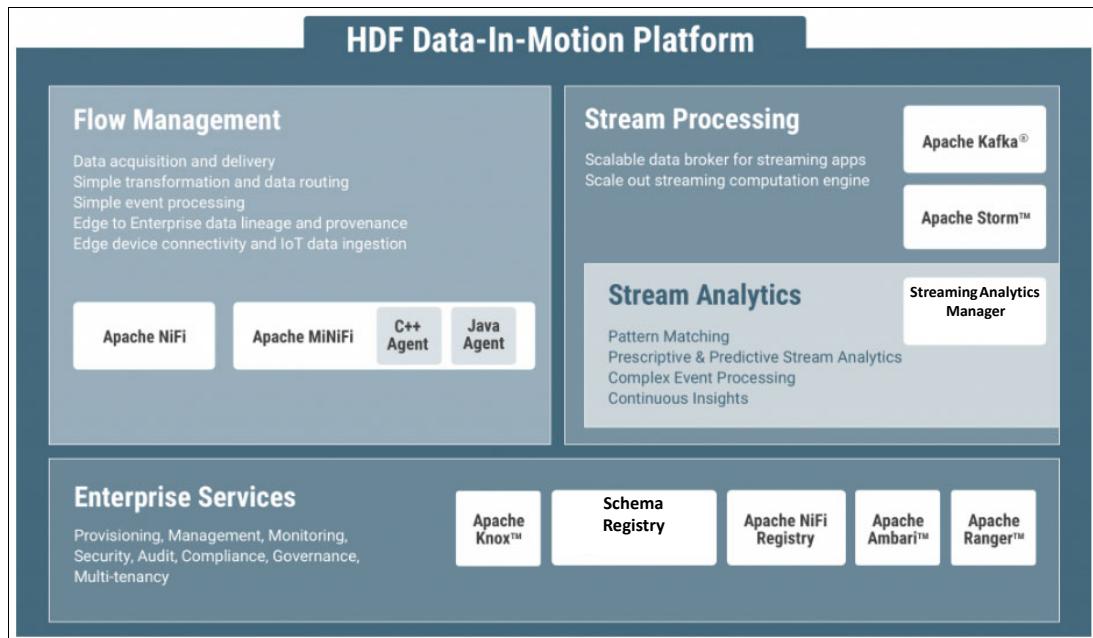


Figure 5-2 Hortonworks DataFlow components

Like HDP, HDF is a cluster of components from the Hadoop Project that is maintained by the Apache Foundation. Many of these components already are integrated to talk to each other by Hortonworks to streamline work and deliver results faster.

This chapter describes each of the main components of HDF for Streaming and Real-time Analytics.

5.1.1 Planning a Hortonworks DataFlow installation

As mentioned earlier, the HDF is composed of some components from the Apache Foundation, and many of them are related to the Hadoop project. HDF can be installed in an existing HDP environment or on a new HDP cluster. It also can be installed independently without a traditional Hadoop solution such as HDP.

Installing Hortonworks DataFlow services on an existing Hortonworks Data Platform cluster

If you already have an existing HDP cluster with Kafka and Storm that is protected by Ranger and Knox and want to use NiFi and MiNiFi on the same cluster, install the HDF Management Pack on the Ambari server, which enables the HDF repositories on the HDP nodes, and then do the push installation of the service (NiFi and MiNiFi).

To install HDF on an existing HDP cluster, complete the following steps:

1. Upgrade Ambari.
2. Upgrade HDP.
3. Install the databases if they do not exist (they are needed for the HDF services).
4. Install the HDF Management Pack on the Ambari server.

5. Update the HDF base URL.
6. Add the HDF services to the existing HDP cluster.

Installing HDF services on a new HDP cluster

This scenario applies if you want to create a cluster of HDP and then add the HDF services NiFi and MiNiFi.

For this scenario, you must determine the entire HDP cluster size depending on your cluster utilization. For help with this sizing, contact your IBM Support Services Representative (IBM SSR) or Hortonworks sales representative.

To install HDF services on a new HDP cluster, complete the following steps:

1. Install Ambari.
2. Install the databases (they are needed for the HDP and HDF services).
3. Install and create an HDP cluster by using the Ambari interface.
4. Install the HDPE Management Pack on the Ambari server.
5. Update the HDF base URL.
6. Add HDF services into the new HDP cluster.

Installing an HDF cluster

Consider a scenario where you want to install, without an HDP cluster, the entire stack from the HDF. This stack has the following components:

- ▶ Operations:
 - Ambari
 - Zookeeper
 - Schema registry
 - NiFi registry
- ▶ Security: Ranger
- ▶ Streaming and integration:
 - NiFi
 - MiNiFi
 - SAM
 - Storm
 - Kafka

To install the HDF cluster, complete the following steps:

1. Install Ambari.
2. Install the databases (they are needed for the HDF services).
3. Install the HDPE Management Pack on the Ambari server.
4. Install an HDF cluster by using Ambari.

5.2 Apache NiFi

There are many open source tools that you can use to ingest data from a plain text or log file in to the Hadoop file system, query data from a relational database or a NoSQL database and then ingest it into the Hadoop file system, or ingest data from almost any source to almost any destination, like a webserver log in to database tables instead of into the Hadoop Distributed File System (HDFS).

Apache NiFi automates the flow of the data between the components of a system to process and distribute data, and secure and track the data in a highly available, loss-tolerant, low-latency, and high-throughput environment.

By using a web-based interface and a flow-based programming model, it is possible to create a flow of the data to deliver an app in few hours or even minutes.

Apache NiFi is part of the HDF and can be installed as a service from the HDF repository. NiFi also can be installed as a stand-alone application.

5.2.1 Adding the Apache NiFi service to an HDF cluster

To add the Apache NiFi service to an HDF cluster or a HDP+HDF cluster, complete the following steps:

1. Access and log in to the Ambari interface.
2. On the Dashboard View, select **Actions** → **Add Service**.
3. Select **NiFi** and **NiFi Registry** in the Add Service wizard window and click **Next**.
4. Select where the NiFi and NiFi registry master services will be installed.
5. In the Assign Slave and Client window, select where the NiFi certificate authority will be installed.
6. In the Customize Service window, select **Encrypt Configuration Master Key Password** and **Sensitive property values encryption password** for the NiFi configurations, and **Encrypt Configuration Master Key Password** for the NiFi registry configurations. (Both passwords must have 12 or more characters.)

The next window is a summary window. Click **Install**, and the installation of Apache NiFi on the HDF and HDP cluster begins.

5.2.2 Working with Apache NiFi

NiFi enables a consumer to receive, route, transform, sort, and send data by using a drag GUI. An Apache NiFi data flow can be changed while the system is running, for example, routing data and loading it in to a new database that previously was not part of the data flow.

Figure 5-3 shows a simple flow of data on the NiFi GUI.

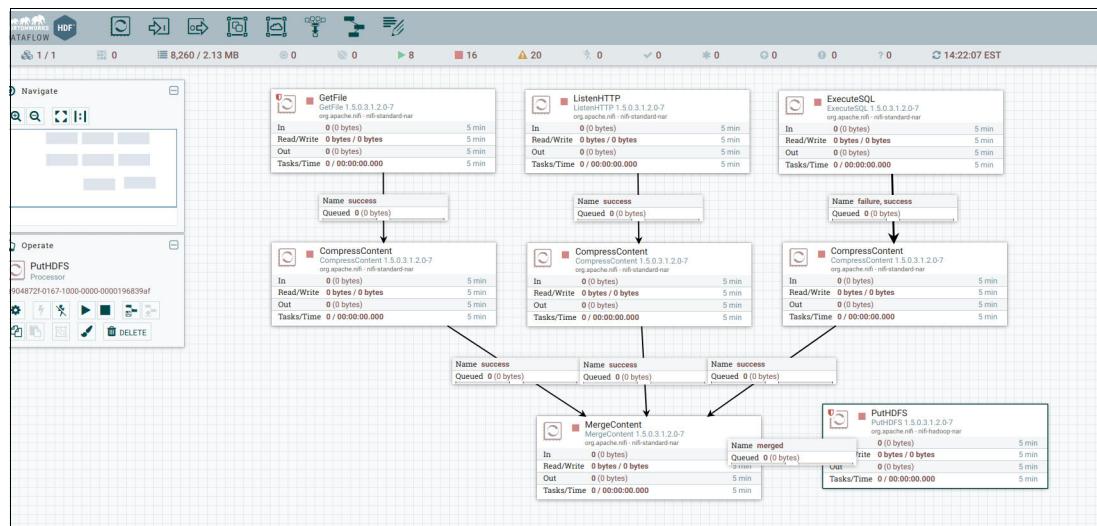


Figure 5-3 Data flow on the NiFi GUI

The data flow on the NiFi is created by block pieces that are called *processors*. Each processor is responsible for one task, such as connecting to a database and running a query, fetching data from an FTP server, loading fields from a JSON file into variables, and inserting a previously loaded variable into another database.

5.2.3 Integrating Apache NiFi with a data science tool environment

NiFi can work with almost any kind of data source and data format. The data can come from and go to an FTP server, an HTTP source, or a UDP stream of data, and can connect to a relational database to extract, transform, and load (ETL) data.

IBM Watson Studio Local works with Jupyter Notebook to connect to various services, such as an Apache Spark cluster running remotely on an IBM PowerAI cluster or a Hadoop cluster.

Using Apache NiFi, you can submit jobs remotely to a Spark Livy cluster or create a processor to process part of the data flow directly on the Spark cluster.

The most common approach to integrate NiFi with an external data science tool such as IBM Watson Studio Local is to create a normal data flow into NiFi and then ingest the data into HDFS or Hive, or a Kafka topic, and then use this data directly on a Jupyter Notebook or other tool.

A data science or Hadoop application is built by using many tools that communicate with each other. This chapter describes some of these tools and presents a use case.

5.3 Apache Storm

Apache Storm is a scalable, fault-tolerant, and distributed computing system that processes a real-time data flow by using Apache Hadoop.

The jobs on Storm are called *topologies*, which are different from the traditional MapReduce jobs that are processed on Apache Hadoop, which can work with batch processing. Topologies are composed of various components that are grouped in a *directed acyclic graph* (DAG). Each component of the DAG uses one or more data flows and propagates one or more new flows of data.

Apache Storm also offers data processing that can reprocess data that was not processed initially; no data is lost.

There are many use cases for Apache Storm, such as ETL replacement, real-time analytics, and dynamic ML computing.

A Storm topology has two main components or entities, which are called *Spouts* and *Bolts*. Together, they produce and consume a stream of *tuples*. Here are the definitions of these terms:

Tuple The data structure (such as an integer number, float, string, or array) that is used by the application.

Spout The entity that is responsible to connect to source of the data streams. It has two modes:

- Unreliable: “Fire and forget”.
- Reliable: Retry if a tuple fails.

Bolt	Uses the streams that flow from one or more spouts and produces new streams. A bolt entity can run functions, filter the input tuple, make joins, query a database, and more.
Stream	The endless flow of tuples that the Storm topology serializes and passes to the next bolt.

Superficially, the topology structure is similar to a MapReduce job. The two main differences are that the Storm topology works with a data flow and runs until it is stopped while a MapReduce job has a defined end.

Figure 5-4 shows the basic topology of a Storm application.

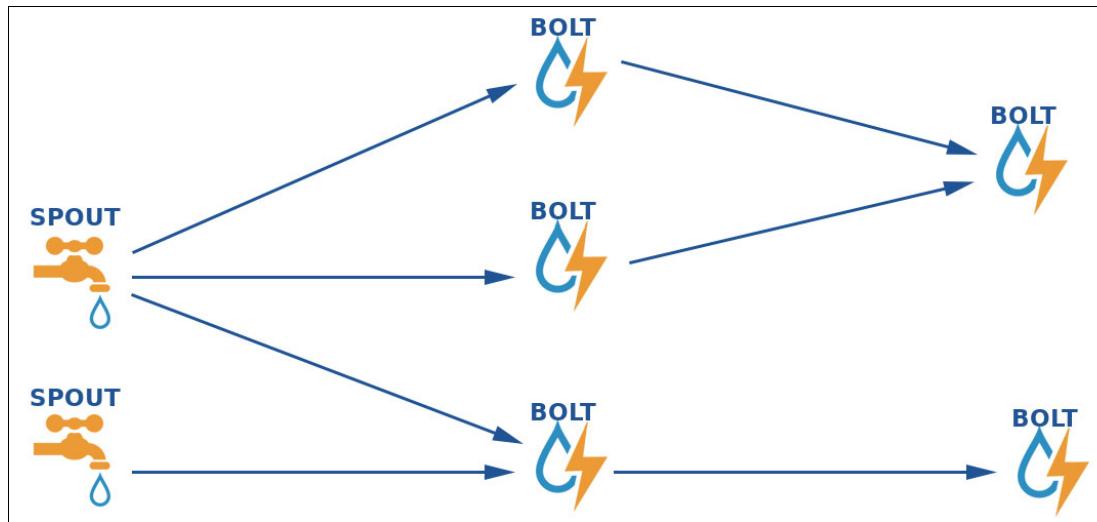


Figure 5-4 Basic Storm application topology

5.3.1 Working with Apache Storm

Apache Storm supports multiple programming languages, that is, each component can be written in a different language. The main language for Storm is Clojure, which is a general-purpose programming language that can be used for multithreaded programming that runs on top of a Java virtual machine (JVM). Storm also supports adapter libraries like Ruby, Python, and Fancy. Any bolt that is *not* written in Clojure runs as a subprocess, which is not the most efficient option.

Unlike Apache NiFi, Storm does not use a GUI, and all the spouts, bolts, and topologies must be written in the same language. This task can be complicated for someone who is not familiar with Java or another programming language, but the gain in performance from using Apache Storm for data-in-motion and a larger flow of data is significant.

5.3.2 Integrating Apache Storm with a data science tool environment

Apache Storm integrates well with many different components of the Hadoop project. This integration is done by a spout, which captures data from a source, a bolt, which can compute the data flowing into a Hadoop's component, and a Storm topology, which ingests the data into another Hadoop component.

Here are some of the components that are supported by Apache Storm:

- ▶ Apache Kafka
- ▶ Apache HBase
- ▶ Apache HDFS
- ▶ Apache Hive
- ▶ Apache Solr
- ▶ Apache Cassandra
- ▶ Apache RocketMQ
- ▶ Java Database Connectivity (JDBC) for general database connection
- ▶ Message Queuing Telemetry Transport (MQTT) for small messages that are frequently used for IoT devices
- ▶ Redis
- ▶ Elasticsearch
- ▶ Mongodb
- ▶ Kinesis
- ▶ Druid
- ▶ YARN
- ▶ Docker
- ▶ Mesos
- ▶ Kubernetes

Because Apache Storm can integrate with most Hadoop components, it also can directly integrate with any data science tool. For example, an ML model running in a Python notebook consumes messages from a Kafka topic, which is loaded by messages that are provided by a bolt from a Storm topology.

5.4 Apache Spark Streaming

Spark Streaming is an extension of the central Spark API that permits scalable flow processing with high performance and fault tolerance. The data can be ingested from many data sources and processed by complex algorithms that are based on high-level functions, such as map, reduce, join, and window. The processed data can be stored on disk (HDFS), where it can be written into a database's table and online dashboards. It is possible to apply a SparkML ML algorithm or even run traditional Spark graph analysis on this live data.

Figure 5-5 shows the flow of the Spark Streaming engine.



Figure 5-5 Flow of the Spark Streaming engine

Spark Streaming receives the high input of data stream, breaks it into small batches that are processed by the Spark engine, and then generates the final output stream in batches.

Inside Spark Streaming, the continuous data flow is represented by a *DStream*. The *DStream* is a high-level basic abstraction for flowing data. A *DStream* can be created from a data tool like Kafka, NiFi, or Flume, or it can be created by an application based on earlier *DStreams*. Internally, the *DStreams* is represented as a sequence of resilient distributed data sets (RDDs).

You can write a Spark Streaming program by using Java, Python, or Scala. The Spark and Spark Streaming program is written in Scala because it is the framework's native language and not all APIs that are implemented for Scala are available in Python.

Beside the *DStream*, there is another important component to be aware for Spark Streaming, which is call *StreamingContext*. It is similar to the *SparkContext* that is used on Spark. *StreamingContext* is the start point for any streaming task that uses Spark Streaming. *StreamingContext* contains internal methods to ingest a continuous flow of data into a Spark Streaming application.

The Spark Streaming API comes with many methods to process data streaming, which are similar to the operations that are used on RDD, such as map, flatMap, filter, count, reduce, groupByKey, reduceByKey, sortByKey, and join. Spark Streaming also provides an extra API for data processing based on window and operations based on state. Those additional methods include window, countByWindow, reduceByKey, reduceByKeyAndWindow, and updateStateByKey.

5.4.1 Working with Apache Spark Streaming

Spark Streaming is a framework API that requires a programming language platform like Java, Scala, or Python to make a program and work with data-in-motion.

The Spark Streaming API is more accessible than the API that is offered by Apache Storm because it does not require a significant amount of effort to learn if you are familiar with the traditional Spark programming paradigm.

Here is a quick example of making a Spark Streaming program to receive data from a server that is listening on a TCP port and then processing this data. This example was taken from the Apache Spark website.

1. Import `SparkContext` and `StreamingContext`, which are the entry points for any streaming method. Instantiate `SparkContext` with two execution threads, and instantiate `StreamingContext` with the two execution threads coming from the `SparkContext` object. Define a batch interval of 1 second.

Example 5-1 shows the Python code that is needed to accomplish this step.

Example 5-1 Python code: Importing and instantiating SparkContext and Streaming Context

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
```

2. With `StreamingContext` defined as `ssc`, create a `DStream` that represents the streaming data that comes from a TCP source that will be defined.

Example 5-2 shows the Python code that is needed to accomplish this step.

Example 5-2 Python code: Creating a DStream

```
lines = ssc.socketTextStream("server.ibm.com", 9999)
```

3. The `lines` `DStreams` object is the stream of data that is received from the defined server. Each record coming from this `DStream` is a line of text. Split these lines into words. (You are making a basic word count program with data-in-motion.)

The delimiter character to split words is a *space*.

Example 5-3 shows the Python code that is needed to accomplish this step.

Example 5-3 Python code: Splitting the lines into words

```
words = lines.flatMap(lambda line: line.split(" "))
```

4. The `flatMap` operation from the `DStream` object creates a `DStream` with multiple records from each record coming from the source `DStream`. Each line is split into multiple words and generates the new `DStream` that is called `words`.

The next step is to count these words.

Example 5-4 shows the Python code that is needed to accomplish this step.

Example 5-4 Python code: Counting the words

```
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)

wordCounts.pprint()
```

5. The `words` `DStream` is then mapped, transformed to a tuple `(word, 1)`, and sent to a new `DStream` that is called `pairs`, which is then reduced to get the count of each word for each `DStream` for the defined batch interval (`ssc = StreamingContext(sc, 1)`) of data, and then loaded into the new `DStream` `wordCounts`.

The `wordCounts.pprint()` function prints the counts of each word that is generated during the time interval.

6. The code that was previously inserted only sets up the computation. To start the Spark Streaming engine and then make the program start capturing the data flowing from the defined TCP Server, call the `StreamingContext.start()` function.

Example 5-5 shows the Python code that is needed to accomplish this step.

Example 5-5 Python code: Starting the `StreamingContext.start()` function

```
ssc.start()  
ssc.awaitTermination()
```

When running the program, any data coming from the server on the defined port is processed by Spark Streaming and then printed in the screen after the defined time window, as shown in Example 5-6.

Example 5-6 Program output

```
-----  
Time: 2018-12-19 11:20:20  
-----  
('spark', 19)  
('streaming', 19)  
('count', 5)  
('ibm', 3)  
('word', 3)
```

5.5 Apache Kafka Streams

To understand Kafka Streams, you must first learn about Apache Kafka.

Apache Kafka is a simple but highly distributed and performant message broker system. The Kafka configuration and topology is simple: The producers send records or messages to the Kafka Cluster, which manages those messages and delivers them to the consumers.

Figure 5-6 shows the basic architecture of Apache Kafka.

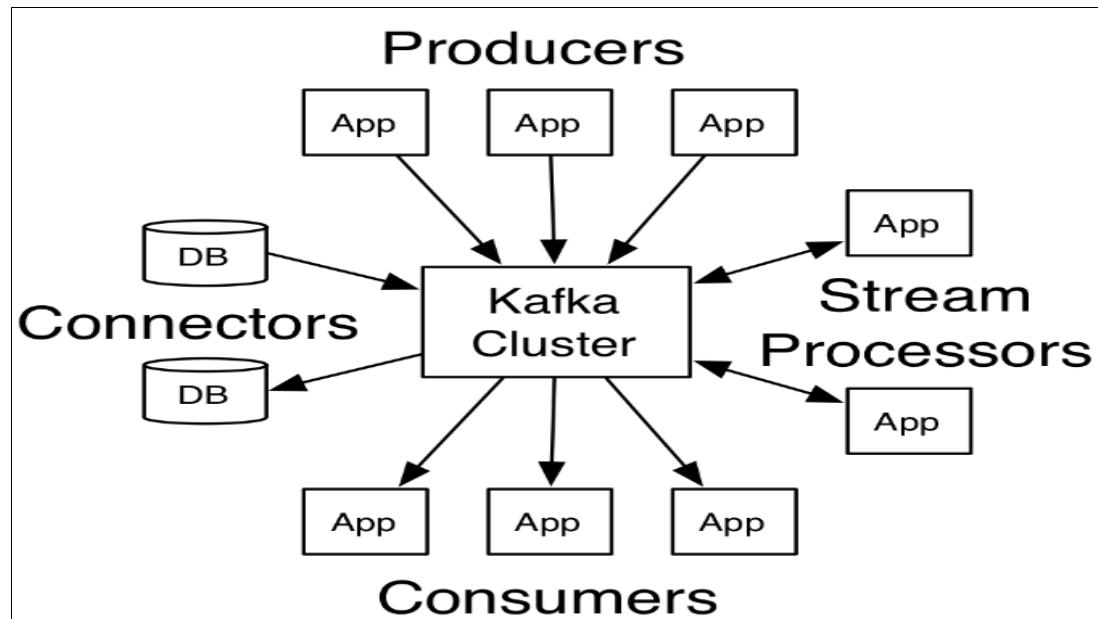


Figure 5-6 Apache Kafka architecture

The key abstraction of Kafka is the *topic*. The producers publish records in a topic and the consumers read from one or more topics. Each record consists of a key, a value, and a time stamp.

A Kafka cluster maintains the topic by using partitions of logs. Each record persists in a slot of a *partition*, and each new record is appended to the top of the partition. Therefore, the partition is an ordered and immutable sequence of records. Each record in the partition is assigned a sequential ID number that is called *offset* that identifies each record.

Figure 5-7 shows the anatomy of a Kafka topic.

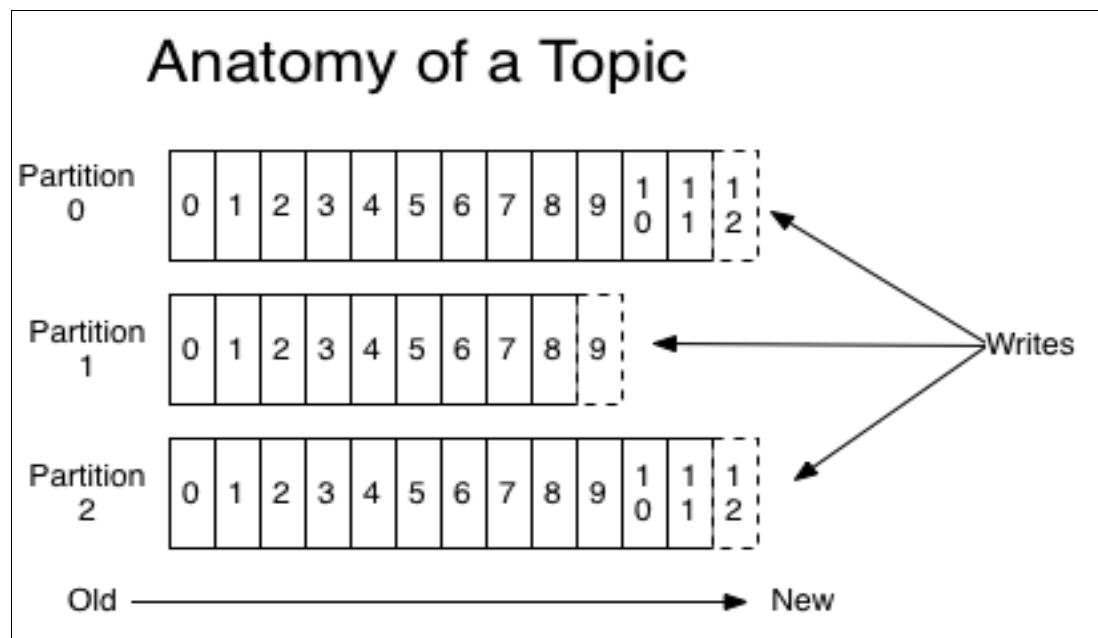


Figure 5-7 Anatomy of a Kafka topic

The Kafka cluster makes all published records persist in the partitions whether or not they are consumed by a consumer. The records remain in the topic for a defined retention period. After the retention period expires, the records are discarded.

Consumers can control the offset position of their topic. Usually, a consumer advances their offset linearly as they read the records from the topic, but it can be consumed in any order. This flexibility to change the offset permits the consumer to reprocess data from a past record.

Figure 5-8 shows how the messages are organized and consumed into a Kafka topic partition.

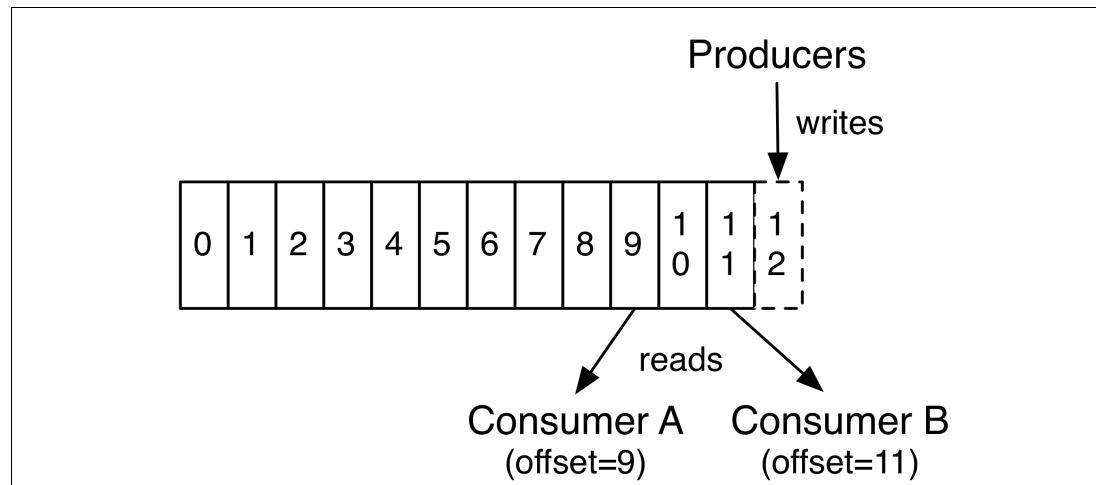


Figure 5-8 The anatomy of a Kafka topic partition

Because the produce and consume processes on topics are lightweight, Kafka can work faster to support up to billions of messages per second in a cluster of just a few brokers.

Kafka Streams is a client library that connects to an Apache Kafka cluster to build applications and microservices. The input and output data that is generated is stored in a Kafka cluster topic. Because Kafka Stream uses an existing Kafka cluster infrastructure, it does not require a cluster configuration.

Figure 5-9 shows how Kafka Streams integrates into a Kafka flow.

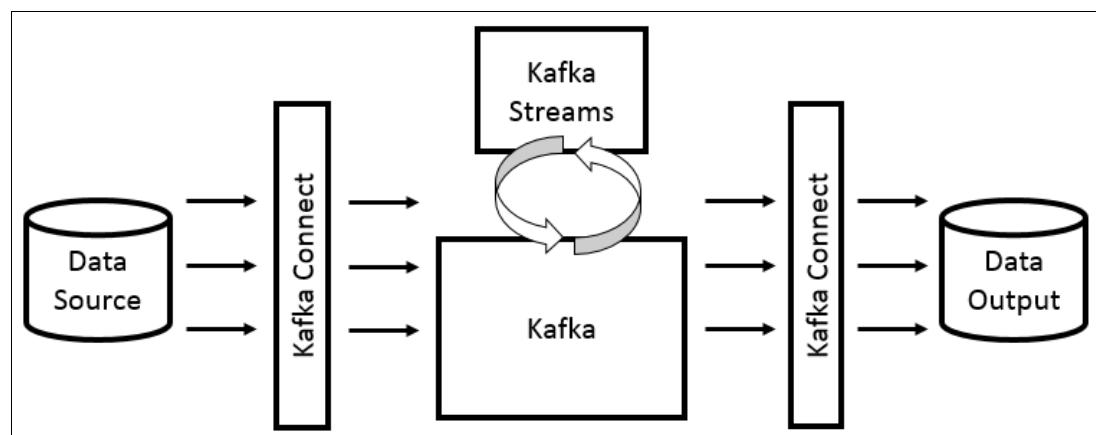


Figure 5-9 Kafka Streams integration into a Kafka flow

The topology of a Kafka Streams program is similar to an Apache Storm topology. They are composed of processing nodes that look like the spouts and bolts of a Storm program. Each processing node can be defined with three different functions:

- ▶ Source processor: The processor that is responsible for consuming the records of a topic and sending them to another processor. The source processor is always the first processor of a Stream topology.
- ▶ Stream processor: The processor that is responsible for computing the data or record that comes from another stream processor or from a source processor. Here is where the programming logic is. It is normal to have multiple stream processors in a single topology.
- ▶ Sink processor: The third processor in a Kafka Stream topology. It is responsible for receiving the data from a stream processor and writing this record into a topic or to any destination (HDFS, database, or others).

Figure 5-10 show the Kafka Streams components and application flow.

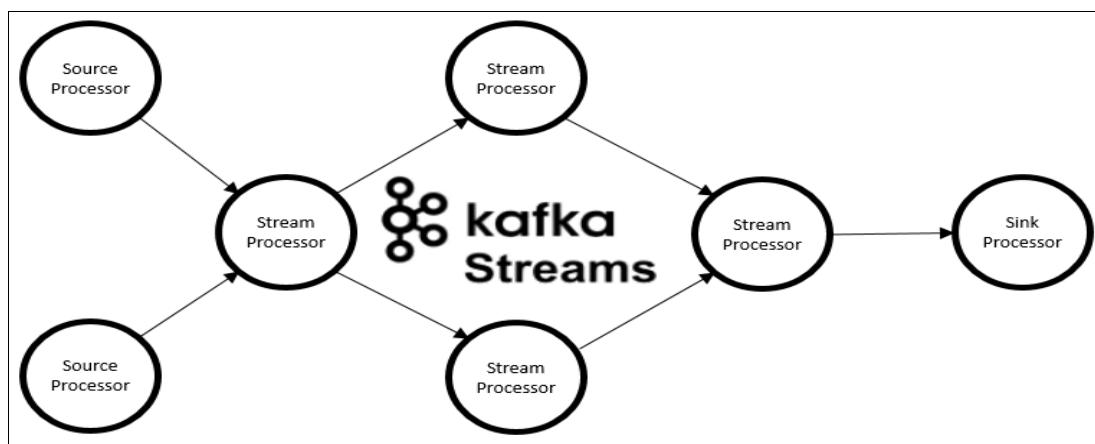


Figure 5-10 Kafka Streams components and application flow

5.6 Integrating streaming tools with data science tools

Now, we show how you can create an application that integrates some HDP and HDF components that are used into a real-time data stream, which can be connected to any data science tool, such as IBM Watson Studio Local or IBM Watson Machine Learning Accelerator.

5.6.1 Application overview

The application has the following functions:

- ▶ Creates a TCP service to stream data on port 9999.
- ▶ Listens on TCP port 9999 for a stream of data by using Spark Streaming.
- ▶ Transfers the data, in batches, to a Kafka topic.
- ▶ Uses the Kafka topic in real time by using NiFi.
- ▶ Merges 10,000 messages into a unique large file by using NiFi.
- ▶ Saves those files into HDFS by using NiFi.

Figure 5-11 shows the flow of the proof of concept (PoC) application that integrates some Hadoop components.

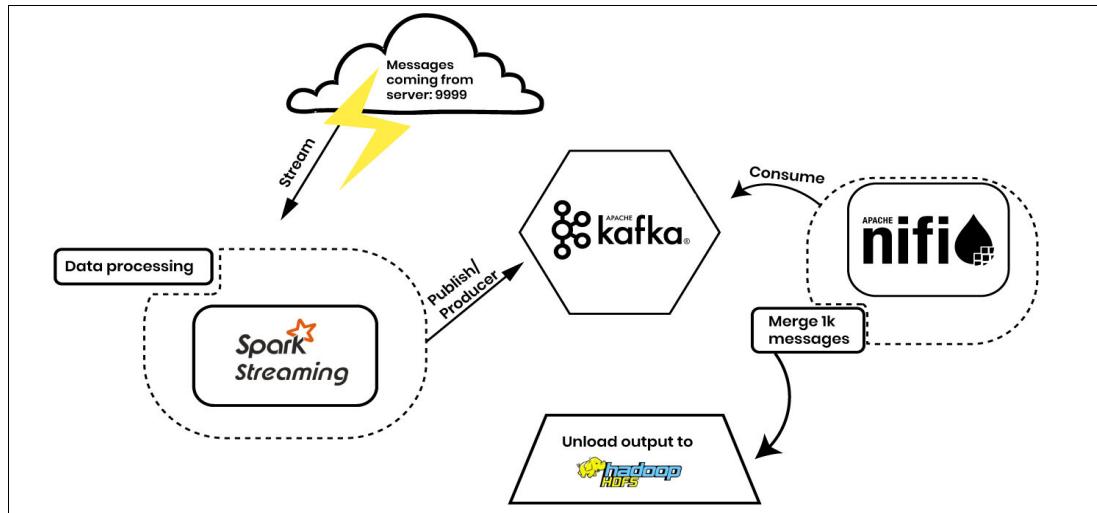


Figure 5-11 Proof of concept application overview

5.6.2 Configuring a Kafka topic

The sample application uses a Kafka topic as a message broker to retain the flow of data that is produced by Spark Streaming.

This section cover the creation and testing of a Kafka topic by using the Kafka command-line interface (CLI). To do so, complete the following steps:

1. Create a topic that is named `kafka-topic-101` by using the `--create` flag on the Kafka CLI, as shown in Example 5-7.

Example 5-7 Linux terminal: Creating a Kafka topic

```
# cd /usr/hdp/current/kafka  
# ./bin/kafka-topics.sh --create --zookeeper zookeeperserver.pbm.ihost.com:2181  
--replication-factor 1 --partitions 1 --topic kafka-topic-101  
Created topic "kafka-topic-101".
```

2. To check the topic, use the `--describe` flag on the Kafka CLI, as shown in Example 5-8.

Example 5-8 Linux terminal: Listing the Kafka topic information

```
# ./bin/kafka-topics.sh --describe --zookeeper  
zookeeperserver.pbm.ihost.com:2181 --topic kafka-topic-101  
Topic:kafka-topic-101PartitionCount:1ReplicationFactor:1Configs:  
Topic: kafka-topic-101Partition: 0Leader: 1001Replicas: 1001Isr: 1001
```

3. To test whether the Kafka topic is working, start a producer, send some messages, and then start the consumer afterward to read all the messages from the topic by using the **--from-beginning** flag, as shown in Example 5-9 and Example 5-10.

Example 5-9 Linux terminal: Producing messages on the Kafka topic

```
# ./bin/kafka-console-producer.sh --broker-list ib-hdprb001.pbm.ihost.com:6667
--topic kafka-topic-101
>message one
>message two
>message three
>it's working good !
>
```

Example 5-10 Linux terminal: Consuming messages on a Kafka topic

```
# ./bin/kafka-console-consumer.sh --bootstrap-server
ib-hdprb001.pbm.ihost.com:6667 --topic kafka-topic-101 --from-beginning
message one
message two
message three
it's working good !
Processed a total of 4 messages
```

5.6.3 Starting a streamer by using the nmap-ncat utility

The **nmap-ncat** utility is a feature-packed networking utility that reads and writes data across networks from the command line. If you do not have the **nmap-ncat** utility already on your server, run **yum** and search for the rpm packet in the Red Hat yum repository, as shown in Example 5-11.

Example 5-11 Linux terminal: yum

```
# yum search nmap-ncat
Loaded plugins: product-id, search-disabled-repos, subscription-manager
=====
N/S matched: nmap-ncat
=====
nmap-ncat.ppc64le : Nmap's Netcat replacement
Name and summary matches only, use "search all" for everything.
```

Complete the following steps:

1. For your real-time data stream application, use `/var/log/messages` as your streamer file. You can use this file by running the **tail -f** command and redirecting the input to a TCP port that uses the **ncat** utility, as shown in Example 5-12.

Example 5-12 Linux terminal: Starting ncat on port 9999 with the output of /var/log/messages

```
# tail -f /var/log/messages | nc -l k 9999
```

2. The **ncat** utility is transmitting the /var/log/messages file in real time to TCP port 9999. Check the service by running **netstat** and test the data stream by running **telnet**, as shown in Example 5-13.

Example 5-13 Linux terminal: Listing the active TCP ports

```
# netstat -nlp | grep 9999
tcp        0      0 0.0.0.0:9999          0.0.0.0:*          LISTEN
12369/nc
tcp6       0      0 :::9999             ::::*              LISTEN
12369/nc
```

The **netstat** utility shows that a process of PID 12369 is *listening* on port 9999 by using the **nc** command.

Example 5-14 Linux terminal: Testing the TCP service with telnet

```
# telnet localhost 9999
Trying ::1...
Connected to localhost.
Escape character is '^'.
Dec 19 14:47:28 ib-hdprb001 xinetd[17566]: EXIT: hdfstransparency status=0
pid=11475 duration=30(sec)
Dec 19 14:47:28 ib-hdprb001 xinetd[17566]: START: hdfstransparency pid=11591
from=10.0.0.79
Dec 19 14:47:28 ib-hdprb001 xinetd[17566]: EXIT: hdfstransparency status=0
pid=11476 duration=30(sec)
Dec 19 14:47:28 ib-hdprb001 xinetd[17566]: START: hdfstransparency pid=11592
from=10.0.0.76
```

5.6.4 Running the Spark Streaming engine

In our application, Apache Spark Streaming collects the flow of data coming from TCP port 9999 of the Streamer server and processes this data by formatting, sorting, and counting it by the type of message that is logged. This formatted data is sent (or published) to the previously created Kafka topic and used by the Apache NiFi processor.

For this task, use Spark Streaming over Python (**pyspark**).

Complete the following steps:

1. In the first lines of code in a Python program, declare which libraries that you want to use, as shown in Example 5-15.

Example 5-15 Python code: Declaring libraries

```
from pyspark.streaming import StreamingContext
from pyspark import SparkContext
from kafka import KafkaProducer
```

- Define a function to write your messages to your previously created Kafka topic, which is named `kafka-topic-101`, as shown in Example 5-16.

Example 5-16 Python code: Defining a function

```
def send_to_kafka_topic(content):
    messages = content.collect()
    for line in messages:
        producer.send('kafka-topic-101', str(record))
    producer.flush()
```

The `send_to_kafka_topic` function is called when a new cycle occurs on `StreamingContext`.

- Define Spark objects in your code to communicate with the Spark server and Kafka Broker server, as shown in Example 5-17.

Example 5-17 Python code: Defining Spark objects

```
# define Kafka Producer connecting to our broker server
producer = KafkaProducer(bootstrap_servers='ib-hdprb001.pbm.ihost.com:6667')

# define SparkContext and StreamingContext
sc = SparkContext(master="local[*]", appName="LogAnalysis-101")
ssc = StreamingContext(sc, 10)
```

The `LogAnalysis-101` log in the `SparkContext` definition is the name of the job to be submitted to Spark. In this case, you are running the code on a master node of the HDP and Spark cluster, which is why the master is pointing to `local[*]`. If you want to connect to a remote Spark cluster, such as an IBM Watson Machine Learning Accelerator Spark Cluster, use `master=spark://remote.master.server:7077` instead.

For the `StreamingContext` definition, the second argument that is passed is the time batch that is processed on Spark Streaming. Spark listens to messages for 10 seconds before it starts processing them. After each 10-second interval, it processes them again.

- Define the DStream. In this example, we name it `stream`, and it defines the connection to the server that is sending messages. In this example, the `ncat` server is already configured, as shown in Example 5-18.

Example 5-18 Python code: Defining the DStream

```
stream = ssc.socketTextStream("ib-hdprb001.pbm.ihost.com", 9999)
```

`Stream` is our Spark Streaming DStream that receives the flow of data every 10 seconds. By using the `.foreachRDD` function, it is possible to process almost any other function on each entry of the RDD. In this case, the entry is a line of the `/var/log/messages` file coming from the Streams server on port 9999. The `.foreachRDD` function works like a loop for each entry on the DStream RDD.

- Run the `send_to_kafka_topic` function to pass each message from our DStream RDD as arguments, as shown in Example 5-19. The `send_to_kafka_topic` function writes those messages to the Kafka topic.

Example 5-19 Python code: Running the send_to_kafka function

```
stream.foreachRDD(send_to_kafka_topic)
```

- Start the Spark Streaming job, as shown in Example 5-20.

Example 5-20 Python code: Starting the Spark Streaming job

```
ssc.start()  
ssc.awaitTermination()
```

This code must be run by using the syntax that is shown in Example 5-21.

Example 5-21 Linux Terminal: Submitting a Spark job by using a Python program

```
# spark-submit \  
--jars spark-streaming-kafka-0-8-assembly_2.11-2.4.0.jar \  
spark-streaming-101.py
```

Figure 5-12 shows the jobs that are submitted to the Spark cluster. Note the LogAnalysis-101 entry.

The screenshot shows the Apache Spark History Server interface. At the top, it displays the version 2.3.0.2.6.5.0-292. Below that, it shows the event log directory as hdfs://spark2-history/, last updated on 2018-12-21 15:49:45, and the client local time zone as America/Sao_Paulo. There is a search bar and a dropdown to show 40 entries. The main table lists various applications with columns: App ID, App Name, Started, Completed, Duration, Spark User, Last Updated, and Event Log. The LogAnalysis-101 entry is highlighted in the table.

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
local-1545414551841	LogAnalysis-101	2018-12-21 15:49:10	2018-12-21 15:49:36	26 s	root	2018-12-21 15:49:36	Download
local-1545246099414	spark-streaming-101	2018-12-19 17:01:38	2018-12-19 17:01:55	17 s	root	2018-12-19 17:01:55	Download
local-1545246062171	spark-streaming-101	2018-12-19 17:01:00	2018-12-19 17:01:25	24 s	root	2018-12-19 17:01:25	Download
application_1545081429839_0001	livy-session-41	2018-12-17 19:35:37	2018-12-17 19:35:56	19 s	livy	2018-12-17 19:35:56	Download
application_1544932175352_0011	livy-session-40	2018-12-17 18:28:45	2018-12-17 18:29:06	21 s	livy	2018-12-17 18:29:06	Download
application_1544932175352_0010	livy-session-39	2018-12-17 18:15:37	2018-12-17 18:16:04	27 s	livy	2018-12-17 18:16:04	Download
application_1544932175352_0008	livy-session-38	2018-12-17 14:50:37	2018-12-17 15:25:03	34 min	admin	2018-12-17 15:25:03	Download
application_1544932175352_0007	livy-session-37	2018-12-17 14:48:46	2018-12-17 14:49:22	36 s	livy	2018-12-17 14:49:22	Download
application_1544932175352_0006	livy-session-36	2018-12-17 12:15:59	2018-12-17 12:16:20	21 s	livy	2018-12-17 12:16:21	Download

Figure 5-12 Spark History Server

5.6.5 Merging data by using NiFi and saving it on HDFS

This section describes how to use NiFi to retrieve and use messages from a Kafka topic, and then send them to a data lake through an HDFS connector.

Apache NiFi has a GUI that you can use to create big data applications by using processors as building blocks that you connect.

In this example, you create the NiFi application by using only three processors:

- ▶ Use messages from the Kafka topic.
- ▶ Wait for 10,000 messages to accumulate, and then create a large file.
- ▶ Unload the file into the HDFS.

To create these processors, complete the following steps:

1. The first process is called GetKafka. Go to the NiFi Interface, click the processor icon, and drag it to the center of the NiFi project, as shown in Figure 5-13.

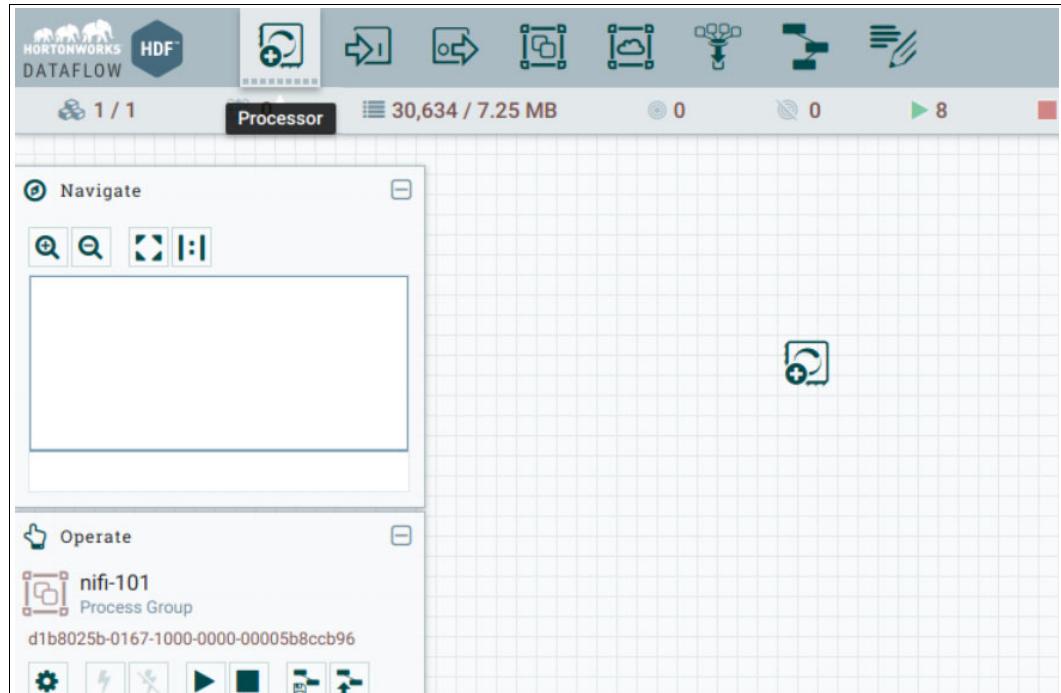


Figure 5-13 NiFi GUI: Adding a processor to the project

2. A window opens. Select the processor that you want to create on the project. Search for GetKafka and click **Add**, as shown in Figure 5-14.

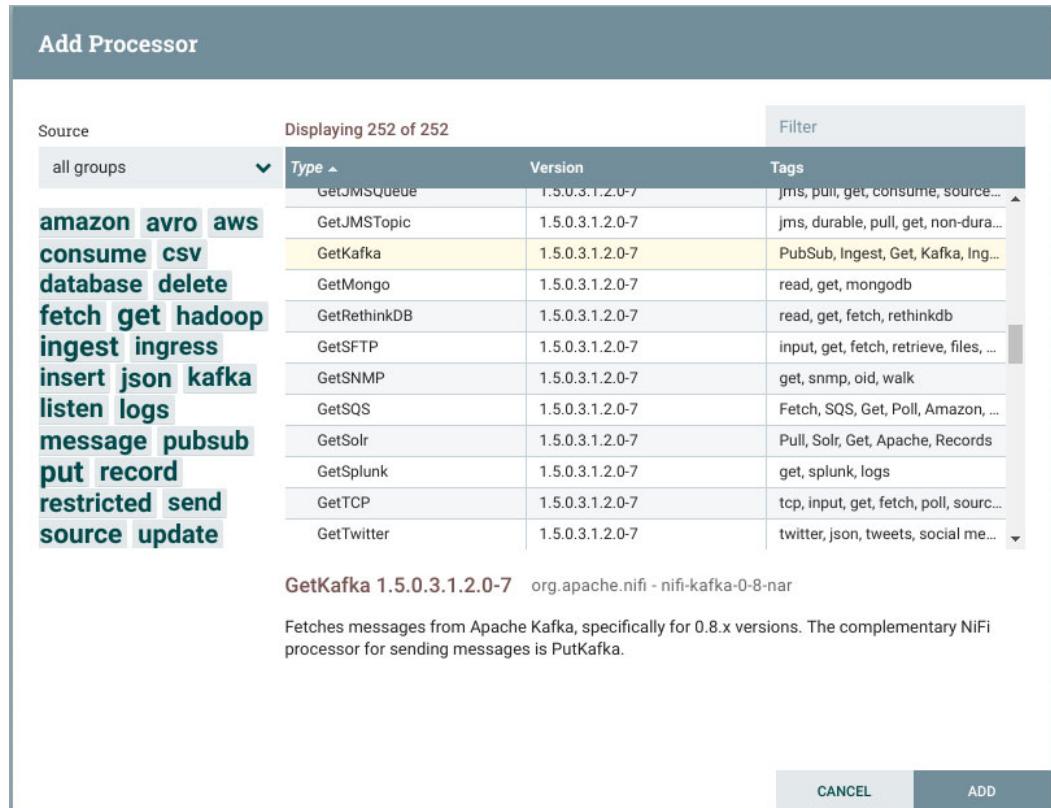


Figure 5-14 NiFi GUI: Selecting the GetKafka processor

3. Add the next two processors by dragging the processor icon to the center of project and selecting the processor that is named MergeContent, as shown in Figure 5-15.

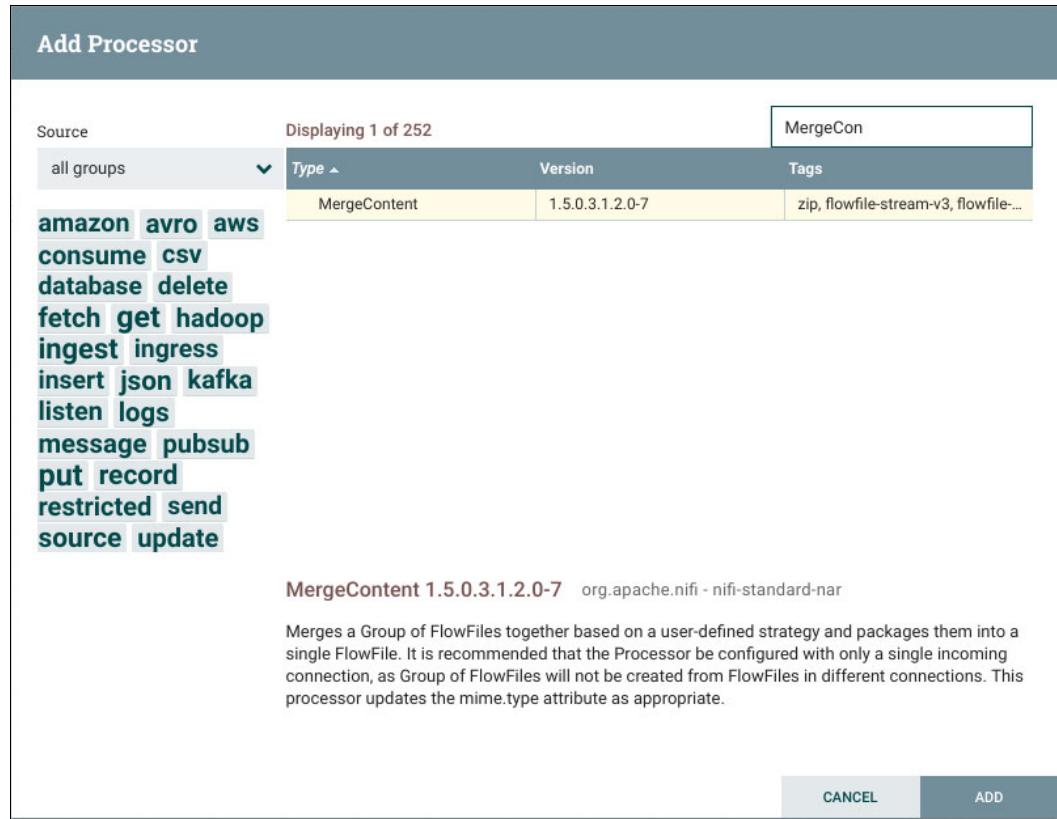


Figure 5-15 NiFi GUI: Selecting the MergeContent processor

- Add the last processor, which is named PutHDFS, as shown in Figure 5-16.

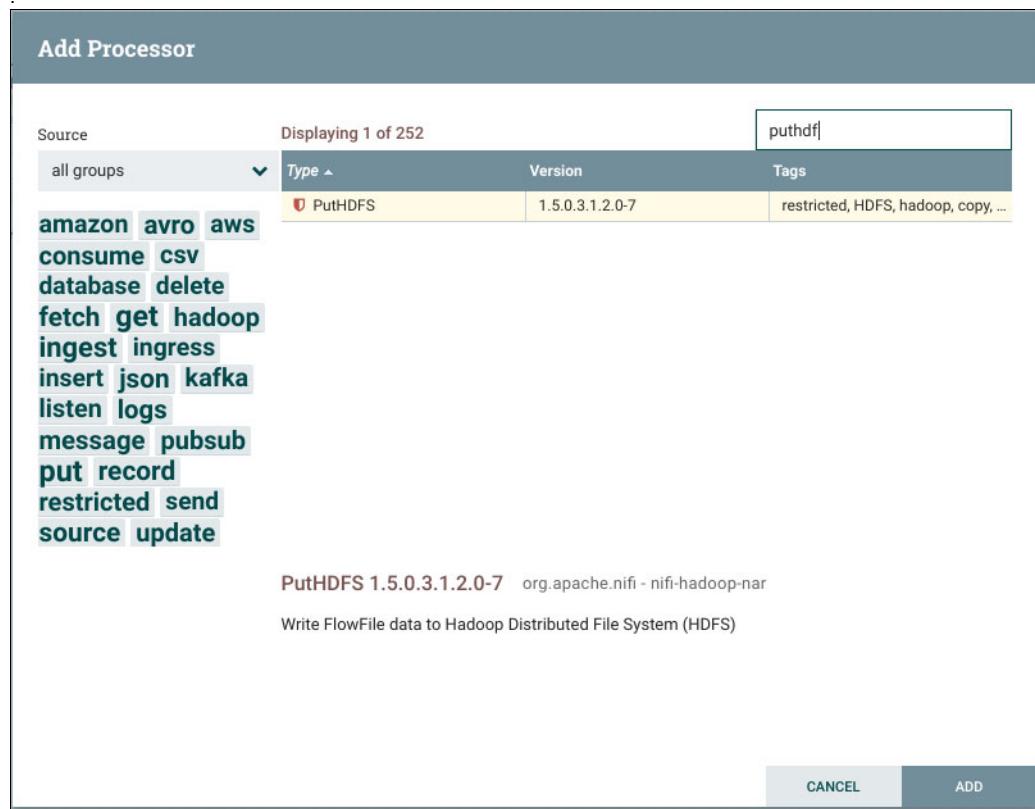


Figure 5-16 NiFi GUI: Selecting the PutHDFS processor

The NiFi application project should look like Figure 5-17.

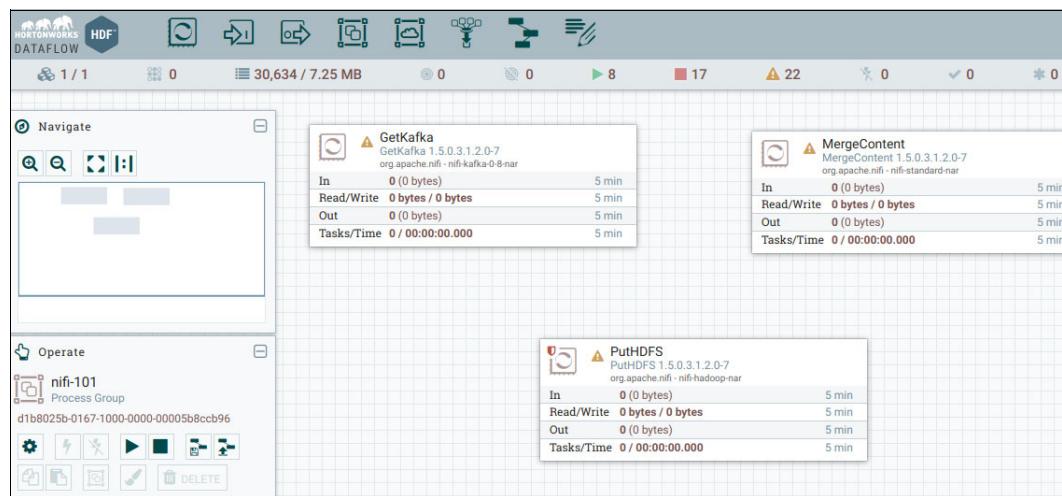


Figure 5-17 NiFi GUI: Application overview

- To connect the processors, hover the mouse over a processor until a dark circle with an arrow appears. Drag the arrow to the next processor to create the flow of the program. Link GetKafka to MergeContent, and then link MergeContent to PutHDFS.

- For every connection that is made between processors, a window opens and asks which content should be passed from one processor to another. For the link between GetKafka and MergeContent, select **Success**. For the link between MergeContent and PutHDFS, select **Merged**.

After the processors are linked, the Project application looks like Figure 5-18.

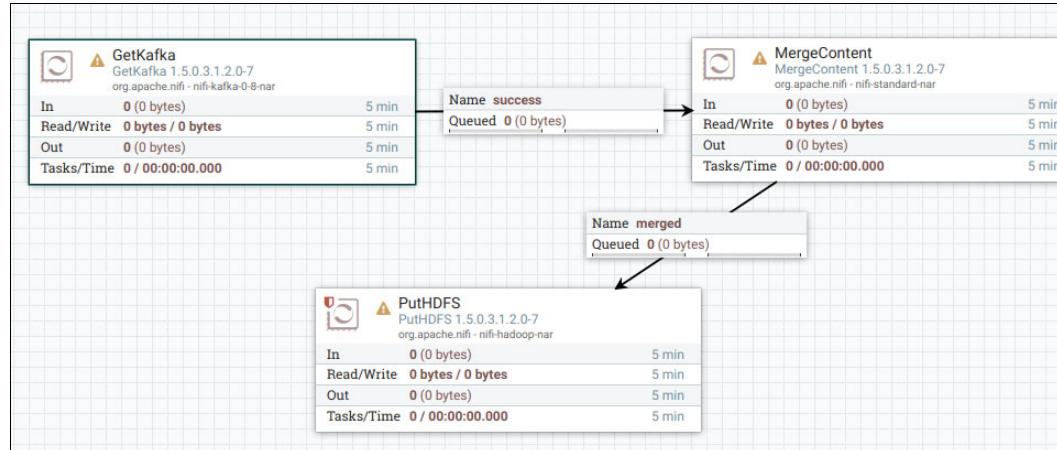


Figure 5-18 NiFi GUI: Linked processors

- Almost every NiFi processor must be configured. To configure your processors, double-click a processor and a window with the properties of the processor opens. For this project, complete the following configurations:
 - GetKafka:
 - The Zookeeper server
 - The Topic Name
 - MergeContent: Minimum Number of Entries
 - PutHDFS:
 - Hadoop Configuration Resources
 - Directory

Figure 5-19 on page 111, Figure 5-20 on page 111, and Figure 5-21 on page 112 show the configuration for each of the three processors.

Configure Processor

SETTINGS	SCHEDULING	PROPERTIES	COMMENTS																						
Required field																									
<table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>ZooKeeper Connection String</td> <td>ib-hdprb001.pbm.ihost.com:2181</td> </tr> <tr> <td>Topic Name</td> <td>kafka-topic-101</td> </tr> <tr> <td>Zookeeper Commit Frequency</td> <td>60 secs</td> </tr> <tr> <td>Batch Size</td> <td>1</td> </tr> <tr> <td>Message Demarcator</td> <td>\n</td> </tr> <tr> <td>Client Name</td> <td>NiFi-d21bdefb-0167-1000-ffff-ffffca33736</td> </tr> <tr> <td>Group ID</td> <td>d21bdefb-0167-1000-ffff-ffffca33736</td> </tr> <tr> <td>Kafka Communications Timeout</td> <td>30 secs</td> </tr> <tr> <td>ZooKeeper Communications Timeout</td> <td>30 secs</td> </tr> <tr> <td>Auto Offset Reset</td> <td>largest</td> </tr> </tbody> </table>				Property	Value	ZooKeeper Connection String	ib-hdprb001.pbm.ihost.com:2181	Topic Name	kafka-topic-101	Zookeeper Commit Frequency	60 secs	Batch Size	1	Message Demarcator	\n	Client Name	NiFi-d21bdefb-0167-1000-ffff-ffffca33736	Group ID	d21bdefb-0167-1000-ffff-ffffca33736	Kafka Communications Timeout	30 secs	ZooKeeper Communications Timeout	30 secs	Auto Offset Reset	largest
Property	Value																								
ZooKeeper Connection String	ib-hdprb001.pbm.ihost.com:2181																								
Topic Name	kafka-topic-101																								
Zookeeper Commit Frequency	60 secs																								
Batch Size	1																								
Message Demarcator	\n																								
Client Name	NiFi-d21bdefb-0167-1000-ffff-ffffca33736																								
Group ID	d21bdefb-0167-1000-ffff-ffffca33736																								
Kafka Communications Timeout	30 secs																								
ZooKeeper Communications Timeout	30 secs																								
Auto Offset Reset	largest																								
CANCEL APPLY																									

Figure 5-19 NiFi GUI: GetKafka processor properties

Configure Processor

SETTINGS	SCHEDULING	PROPERTIES	COMMENTS																														
Required field																																	
<table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Merge Strategy</td> <td>Bin-Packing Algorithm</td> </tr> <tr> <td>Merge Format</td> <td>Binary Concatenation</td> </tr> <tr> <td>Attribute Strategy</td> <td>Keep Only Common Attributes</td> </tr> <tr> <td>Correlation Attribute Name</td> <td>No value set</td> </tr> <tr> <td>Metadata Strategy</td> <td>Do Not Merge Uncommon Metadata</td> </tr> <tr> <td>Minimum Number of Entries</td> <td>10000</td> </tr> <tr> <td>Maximum Number of Entries</td> <td>1000</td> </tr> <tr> <td>Minimum Group Size</td> <td>0 B</td> </tr> <tr> <td>Maximum Group Size</td> <td>No value set</td> </tr> <tr> <td>Max Bin Age</td> <td>No value set</td> </tr> <tr> <td>Maximum number of Bins</td> <td>5</td> </tr> <tr> <td>Delimiter Strategy</td> <td>Filename</td> </tr> <tr> <td>Header</td> <td>No value set</td> </tr> <tr> <td>Footer</td> <td>No value set</td> </tr> </tbody> </table>				Property	Value	Merge Strategy	Bin-Packing Algorithm	Merge Format	Binary Concatenation	Attribute Strategy	Keep Only Common Attributes	Correlation Attribute Name	No value set	Metadata Strategy	Do Not Merge Uncommon Metadata	Minimum Number of Entries	10000	Maximum Number of Entries	1000	Minimum Group Size	0 B	Maximum Group Size	No value set	Max Bin Age	No value set	Maximum number of Bins	5	Delimiter Strategy	Filename	Header	No value set	Footer	No value set
Property	Value																																
Merge Strategy	Bin-Packing Algorithm																																
Merge Format	Binary Concatenation																																
Attribute Strategy	Keep Only Common Attributes																																
Correlation Attribute Name	No value set																																
Metadata Strategy	Do Not Merge Uncommon Metadata																																
Minimum Number of Entries	10000																																
Maximum Number of Entries	1000																																
Minimum Group Size	0 B																																
Maximum Group Size	No value set																																
Max Bin Age	No value set																																
Maximum number of Bins	5																																
Delimiter Strategy	Filename																																
Header	No value set																																
Footer	No value set																																
CANCEL APPLY																																	

Figure 5-20 NiFi GUI: MergeContent processor properties

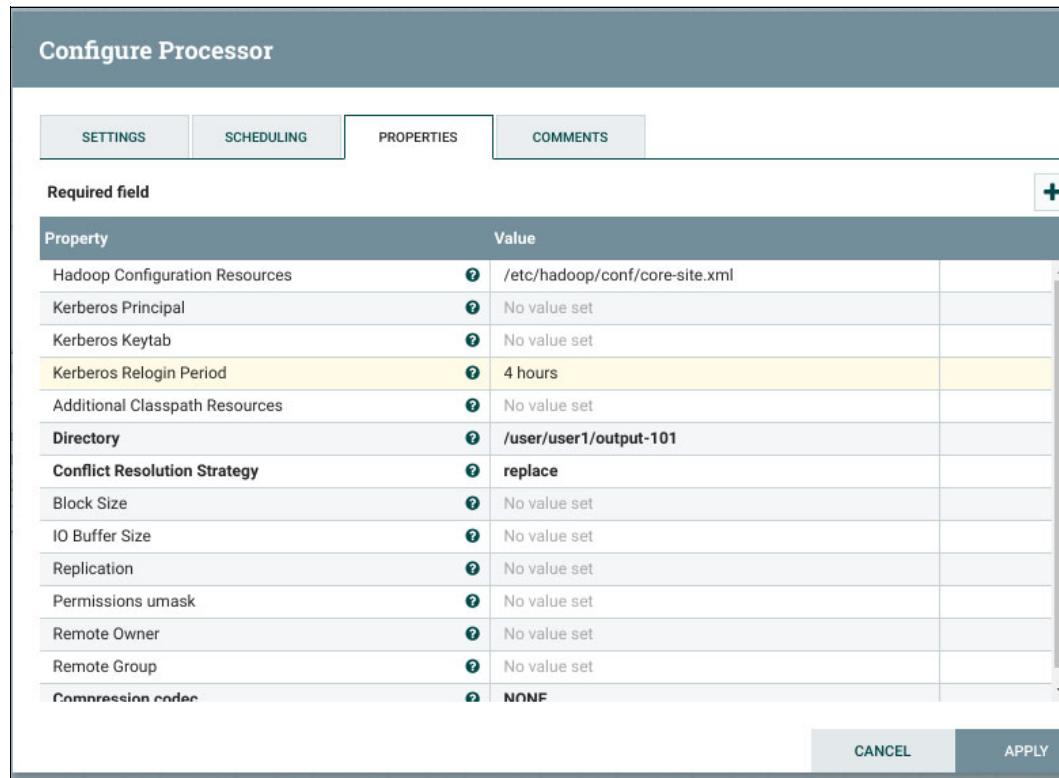


Figure 5-21 NiFi GUI: PutHDFS processor properties

8. You defined which content is passed by one processor to another, but the processors must be configured further. In the Configure Processor window, click the **Settings** tab, and select the **Automatically Terminate Relationship** check boxes for the previously non-selected content, as shown in Figure 5-22 on page 113 and Figure 5-23 on page 114.

For MergeContent, select:

- **Failure**
- **Original**

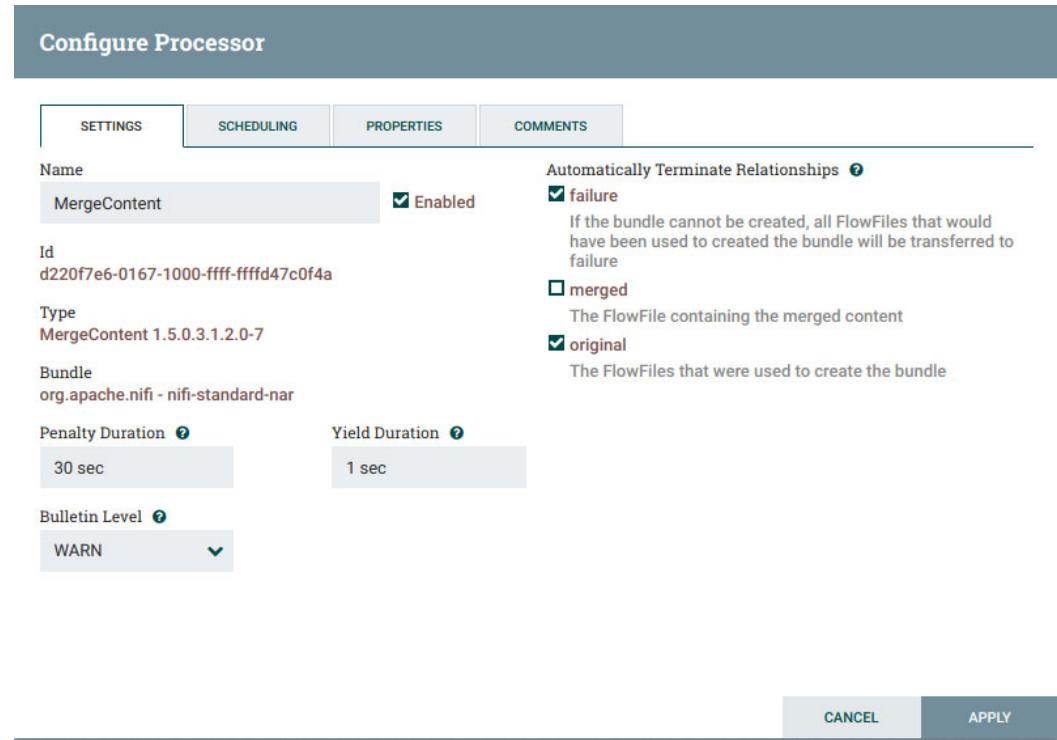


Figure 5-22 NiFi GUI: MergeContent processor settings

For PutHDFS, select:

- **Failure**
- **Success**

Configure Processor

SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
Name PutHDFS	Enabled	Automatically Terminate Relationships  <input checked="" type="checkbox"/> failure Files that could not be written to HDFS for some reason are transferred to this relationship <input checked="" type="checkbox"/> success Files that have been successfully written to HDFS are transferred to this relationship	
Id d225639d-0167-1000-0000-000014db99a3			
Type PutHDFS 1.5.0.3.1.2.0-7			
Bundle org.apache.nifi - nifi-hadoop-nar			
Penalty Duration 	Yield Duration 		
30 sec	1 sec		
Bulletin Level 			
WARN			
<input type="button" value="CANCEL"/> <input type="button" value="APPLY"/>			

Figure 5-23 NiFi GUI: PutHDFS processor settings

9. After every processor is configured, start the first processor, GetKafka, to see the messages that are populating the queue between the GetKafka and MergeContent processors. To start a processor, right-click it and select **Start**, as shown in Figure 5-24.

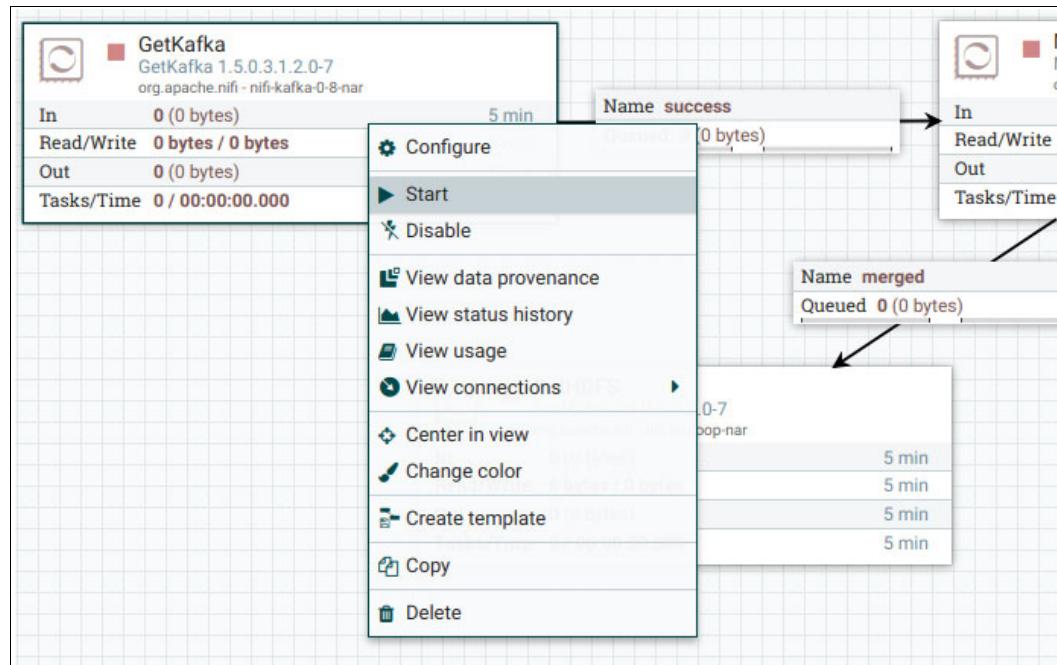


Figure 5-24 NiFi GUI: Starting the first data flow processor

After a few seconds, the messages start to flow from the GetKafka processor to the next processor, which is not started yet. The messages stay and wait on a queue between the processors, as shown in Figure 5-25.

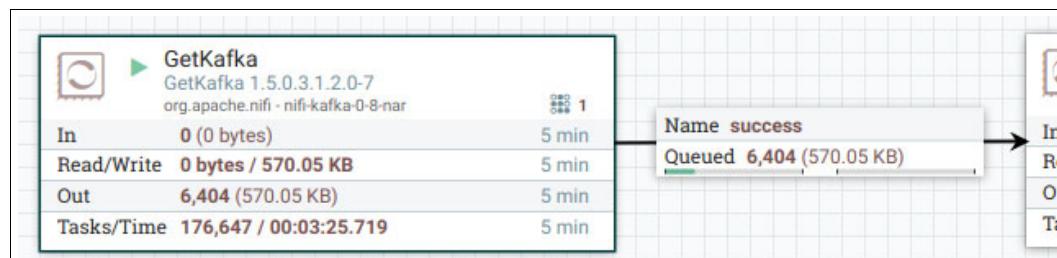


Figure 5-25 NiFi GUI: Checking queued messages

10. Start the MergeContent processor, which takes 10,000 messages that pass from GetKafka and merges them into one large file.

After MergeContent starts and the messages are received by the GetKafka processor, the 10,000 messages in the queue between MergeContent and PutHDFS should increase, as shown in Figure 5-26.

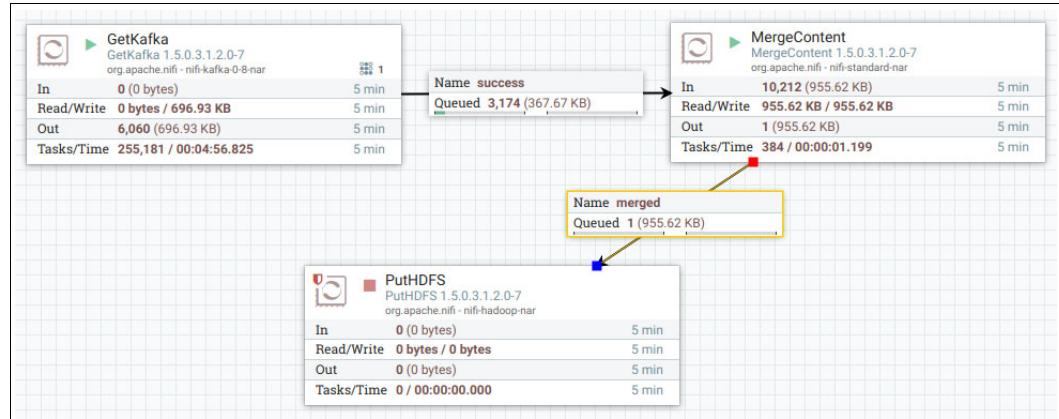


Figure 5-26 NiFi GUI: Merged file on the queue between the MergeContent and PutHDFS processors

11. Starting the last processor unloads the merged file into the HDFS. This file can be used by another application to do an ML task, analytics, or another type of insight.
12. To start the processor, run the command that is shown in Example 5-22.

Example 5-22 Linux terminal: Listing data that is stored on HDFS

```
# hadoop fs -ls /user/user1/output-101
Found 1 items
-rw-r--r-- 3 nifi hadoop 978554 2018-12-21 15:19
/usr/user1/output-101/1999736852385173
```

5.6.6 Conclusion of the data stream integration proof of concept

For this PoC, you integrated the following components from Apache Hadoop:

- ▶ The /var/log/messages and the **nmap-ncat** function
- ▶ Spark / **pyspark**
- ▶ Spark Streaming
- ▶ Kafka
- ▶ NiFi
- ▶ HDFS

You did not transform the data during the Spark part. The Spark and Spark Streaming code only listened to the messages flowing from a server and then loaded these messages to a Kafka topic. There are many transformations that can be done on the data before publishing it to Kafka. There are many sources that can be used to obtain the data instead of using a dummy server with the /var/log/messages content. Apache Spark Streaming is useful for consuming messages from a Kafka topic, transforming those messages, and publishing them to another Kafka topic or even to another Kafka server.

There are many configurations that can be made by connecting the Apache Hadoop components that are provided by HDP and HDF. These components can be and commonly are combined with other non- Hadoop components like ML and deep learning (DL) tools, relational and non-relational databases, and web servers.



A

Additional information

This appendix provides additional information about the lab system setup that was used to create this publication. It is suitable for test work, but does not represent a typical client implementation.

The following topics are covered in this appendix:

- ▶ System topology
- ▶ Software levels

System topology

The basic system configuration is shown in Figure A-1.

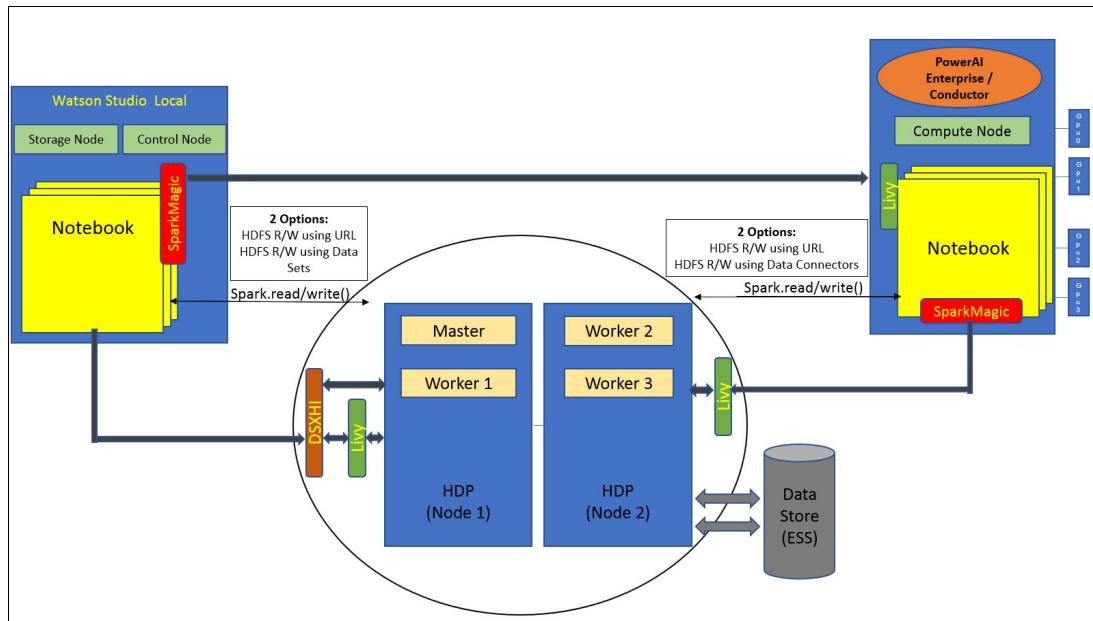


Figure A-1 System topology

Software levels

Table A-1 is related to the Power Architecture and is a collection of all the software stacks in this publication's environment.

Table A-1 Software levels recommendations

Software	OS level	NVIDIA GPU driver	Apache Spark version	NVIDIA CUDA	NVIDIA CUDA Deep Neural Network (cuDNN)	IBM Spectrum Scale
Hortonworks Data Platform (HDP) 3.0	RHEL Maipo7.5	410.72	2.3.0.2.6. 5.0-292	10	7.3.1	5.0.2-0
Hortonworks DataFlow (HDF) 3.1.2.0	RHEL 7.5 Maipo	410.72	2.3.0.2.6. 5.0-292	10	7.3.1	5.0.2-0
IBM Watson Machine Learning Accelerator V1.1.2	RHEL 7.5 Maipo	410.72	2.3.0.2.6. 5.0-292	10	7.3.1	5.0.2-0
IBM Data Science Experience (IBM DSX) Local V1.2.1.0 (20181110_1424) ppc64le	RHEL 7.5 Maipo	410.72	2.3.0.2.6. 5.0-292	10	7.3.1	5.0.2-0
H2O Driverless AI 1.4.2	RHEL 7.5 Maipo	410.72	2.3.0.2.6. 5.0-292	9.0	7.3.1	5.0.2-0



B

Installing an IBM Watson Machine Learning Accelerator notebook

The examples in this appendix help you customize a notebook package, add a notebook package, create a Spark Instance Group (SIG) that includes the notebook package, start a notebook service, and test a notebook.

The base notebook package and some sample notebooks are available [GitHub](#).

The following topics are covered in this chapter:

- ▶ Customizing a notebook package
- ▶ Adding a notebook package
- ▶ Creating a Spark Instance Group with a notebook
- ▶ Creating notebooks for users
- ▶ Testing notebooks

Customizing a notebook package

In this section, we customize a notebook to include Anaconda. That module is used in a Jupyter Notebook to connect to a Livy server running on a Hadoop cluster. Once connected, Spark jobs can be run remotely.

Start by downloading the PowerAI-1.5.4.1-Notebook-Base.tar.gz notebook package from [Github](#) to the workstation where you are running the IBM Watson Machine Learning Accelerator Cluster Management Console.

The PowerAI-1.5.4.1-Notebook-Base.tar.gz package includes the scripts that are used to start, stop, and monitor the notebook server.

This section shows how to update the PowerAI-1.5.4.1-Notebook-Base.tar.gz package to create a PowerAI-1.5.4.1-Notebook-HI.tar.gz package. A newer version of the notebook package might be available, such as Version 1.5.4.2. If so, use the newer version in the following instructions by replacing V1.5.4.1 with the latest version.

1. Create a custom working directory and extract the PowerAI-1.5.4.1-Notebook-Base.tar.gz package into it by running the following commands:

```
mkdir custom
tar -C custom -xzvf PowerAI-1.5.4.1-Notebook-Base.tar.gz
```
2. Download the Anaconda repo file by running the following commands:

```
cd custom/package
wget https://repo.continuum.io/archive/Anaconda3-5.2.0-Linux-ppc64le.sh
cd ..
```
3. (Optional) Update the build.version to include today's date. This example shows the edits by using a vi editor.

```
vi build.version (or use an editor of your choice)
```

 - a. Enter insert mode (a) and edit the build data and number. For example:

```
Build Date: "Jan 17 2019"
Build Number: 20190117
```
 - b. Press Esc and save the update (:qw).
4. (Optional) Add more pip packages of interest for your custom notebook by updating the added_packs file:

```
vi scripts/added_packs
```

 - a. Enter insert mode (a) and scroll down to bottom to add additional packages (one per line):

```
theano
keras
pandas
sparkmagic
tensorvision
```
 - b. Press Esc and save the update (:qw).
5. Compress the updated custom package (HI = Hadoop Integration) by running the tar command:

```
tar -czvf PowerAI-1.5.4.1-Notebook-HI.tar.gz
```

In the next section, we add this notebook package to IBM Watson Machine Learning Accelerator.

Adding a notebook package

This section describes how to add the notebook package that was customized in “Customizing a notebook package” on page 122 into IBM Watson Machine Learning Accelerator.

To add the notebook package, complete the following steps:

1. Using the management console, go to the notebook management user interface by selecting **Workload** → **Spark** → **Notebook Management**, as shown in Figure B-1.

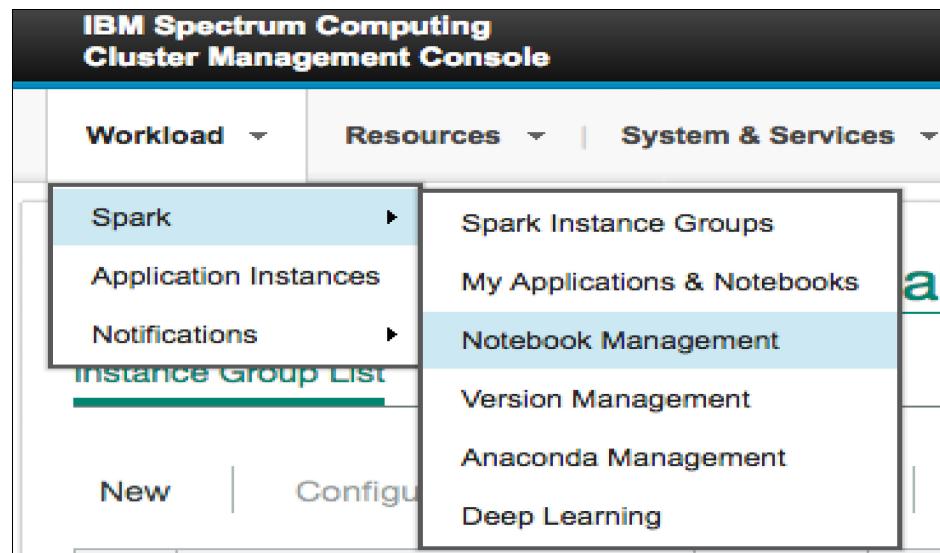


Figure B-1 Notebook Management menu

2. Click **Add**, as shown in Figure B-2.

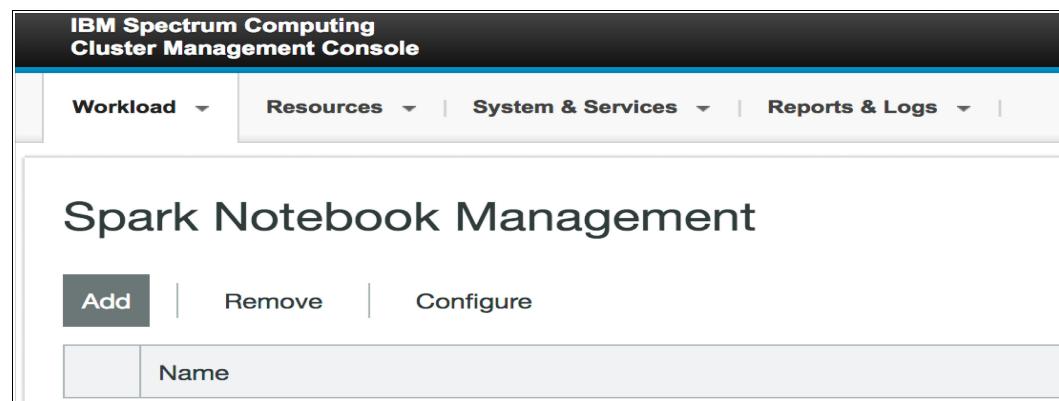


Figure B-2 Notebook Management user interface

3. Complete the details for the PowerAI-1.5.4.1-Notebook-HI.tar.gz notebook and click **Add**, as shown in Figure B-3:
 - a. Give it a name (PowerAIHI) and a version (1.5.4.1 or match the version of the notebook .tar file).
 - b. Click **Browse** to find the tar.gz file that you downloaded or customized.
 - c. Select the **Enable monitoring for the notebook** and **Enable collaboration for the notebook** check boxes.
 - d. Complete the Start command, Stop command, and Job monitor command fields as follows:
 - ./scripts/start_jupyter.sh
 - ./scripts/stop_jupyter.sh
 - ./scripts/jobMonitor.sh
 - e. In the Longest update interval for job monitor field, specify the number of seconds (180 is a good default).

Add Notebook

(i) To auto-populate the fields, drag either *.yml or <notebook_name>-<version>.tar.gz, or both files.

Deployment Settings **Environment Variables**

* Name:	PowerAIHI
* Version:	1.5.4.1
* Package:	<input type="button" value="Browse..."/> PowerAI-1.5.4.1-Notebook-HI.tar.gz
<input type="checkbox"/> Run notebook in a Docker container <input type="checkbox"/> Enable monitoring for the notebook <input checked="" type="checkbox"/> Enable collaboration for the notebook <input type="checkbox"/> Enable SSL support <input type="checkbox"/> Anaconda required	
Prestart command:	
* Start command:	./scripts/start_jupyter.sh
* Stop command:	./scripts/stop_jupyter.sh
* Job monitor command:	./scripts/jobMonitor.sh
Base port:	
Longest update interval for job monitor: <input type="text" value="180"/> seconds	
Job control wait period:	

Add **Cancel**

Figure B-3 Add Notebook wizard

Wait for the copying the process to complete, as shown in Figure B-4. The copy speed depends on the network speed and the package size (over 300 MB in this case).

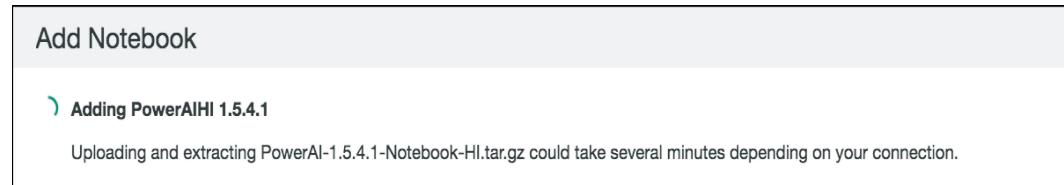


Figure B-4 Notebook copy in progress

After the Add Notebook process is complete, you can add the notebook to an existing SIG by stopping it and updating its configuration, or you can create a SIG and add the notebook to it.

Creating a Spark Instance Group with a notebook

To create a SIG with a notebook enabled, complete the following steps in the IBM Watson Machine Learning Accelerator cluster management console:

1. Select **Spark** → **Spark Instance Groups**, as shown in Figure B-5.

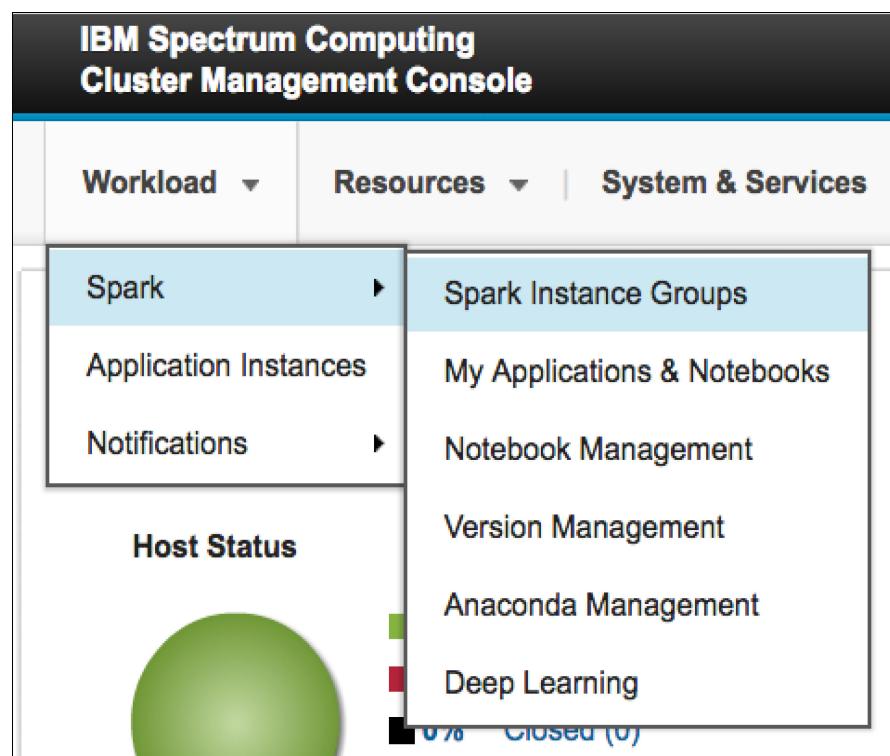


Figure B-5 Spark Instance Groups menu

- Click **New**, as shown in Figure B-6.

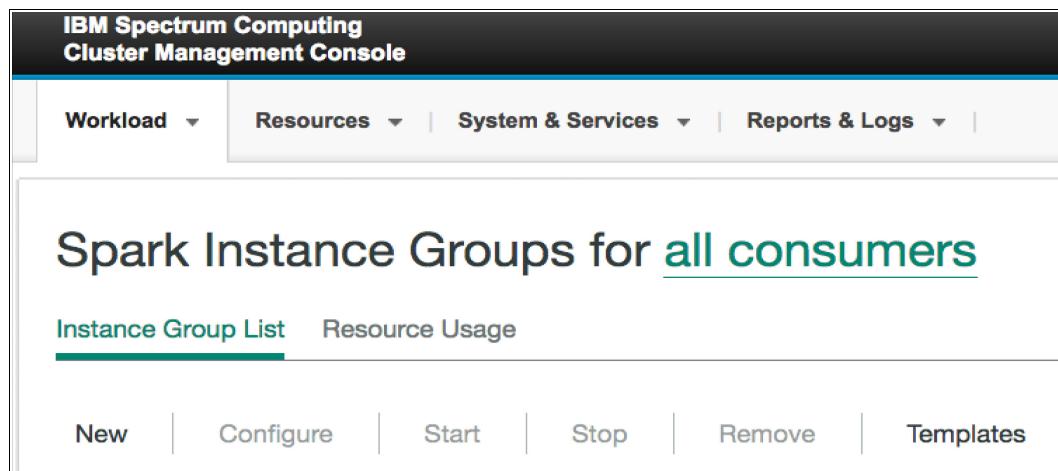


Figure B-6 Creating a Spark Instance Group

- Complete the SIG details. Name the SIG, specify a deployment directory that is on the local storage, and specify the OS user for the SIG, as shown in Figure B-7.

The screenshot shows the 'New Spark Instance Group' configuration page. The top navigation bar is identical to Figure B-6. The main title is 'New Spark Instance Group'. Below the title, there are four tabs: 'Basic Settings' (selected), 'Containers' (Optional), 'Packages' (Optional), and 'Data Connectors' (Optional). The 'Basic Settings' tab contains a section titled 'Required Fields' with three input fields: 'Instance group name' (set to 'Notebook'), 'Spark deployment directory' (set to '/cwslocal/demouser/notebook'), and 'Execution user for instance group' (set to 'demouser').

Figure B-7 New Spark Instance Group: Part 1

We create demouser as the execution user for our tests. IBM Spectrum Conductor Deep Learning Impact template-based SIGs run as egoadmin. For the Spark deployment directory, we append a local directory, a run-as user, and the SIG name (/cwslocal + /demouser + /notebook).

4. Select the check box on the notebook to enable in the SIG, as shown in Figure B-8.

The screenshot shows the 'New Spark Instance Group' configuration interface. In the 'Enable notebooks' section, there are three entries:

- Jupyter 5.4.0 [Collaboration enabled]
- PowerAI 1.5.4.1.1 [Collaboration enabled]
Base data directory: /cwsshare/demouser/notebook
- PowerAI 1.5.4.1 [Collaboration enabled]

Each entry includes a 'Configuration' link below it.

Figure B-8 New Spark Instance Group: Part 2

For the consumers, we used the default that is shown in Figure B-9.

The screenshot shows the 'New Spark Instance Group' configuration interface. In the 'Consumers' section, there are two main sections:

- Spark instance group:** /Notebook
Edit detailed consumer assignments
- Security:**
 - Enable authentication and authorization for the submission user
 - Enable impersonation to have Spark applications run as the submission user

Below the security section, there is a 'User name for resource requests:' field with a dropdown menu labeled 'Select or search for an impersonation user'.

Figure B-9 New Spark Instance Group: Part 3

For the Resources Groups, we specified the compute host resource group (RG) except for the GPU executors, as shown in Figure B-10.

Resource Groups and Plans

Select the resource groups or plans that provide resources to the instance group. [\(i\)](#)

Spark drivers:	ComputeHosts
Spark executors (CPU slots):	ComputeHosts
Spark executors (GPU slots):	GPUHosts
Spark shuffle service:	(i) The CPU executors resource group is used for the shuffle service. The CPU executors resource group must contain all hosts that are in the GPU executors resource group.
Spark master service for batch:	ComputeHosts
Spark master service for notebooks:	ComputeHosts
PowerAIHI 1.5.4.1	ComputeHosts

Create and Deploy Instance Group **Create Only** **Cancel**

Figure B-10 New Spark Instance Group: Part 4

Click **Create and Deploy Instance Group** to create and deploy the SIG.

5. Start the SIG when its status is Ready, as shown in Figure B-11.

The screenshot shows the 'IBM Spectrum Computing Cluster Management Console' interface. The top navigation bar includes 'Workload', 'Resources', 'System & Services', and 'Reports & Logs'. Below the navigation is a section titled 'Spark Instance Groups for all consumers'. Underneath, there are two tabs: 'Instance Group List' (which is selected) and 'Resource Usage'. A horizontal toolbar below the tabs contains buttons for 'New', 'Configure', 'Start' (which is highlighted in dark grey), 'Stop', 'Remove', and 'Templates'. A table follows, displaying two entries:

	Name ↑	State	Spark Version	Updates	Running Applications	CPU Slots
<input type="checkbox"/>	DLI	Started	2.2.0	Up to date	0	4
<input checked="" type="checkbox"/>	Notebook	Ready	2.3.1	Up to date	0	0

At the bottom of the table, it says 'Showing 1 to 2 of 2 entries'.

Figure B-11 Starting a Spark Instance Group

After the SIG starts, notebooks can be created for the users.

Creating notebooks for users

To create notebooks for users, complete the following steps:

1. Go to the **Notebooks** tab in the SIG details and click **Create Notebooks for Users**, as shown in Figure B-12.

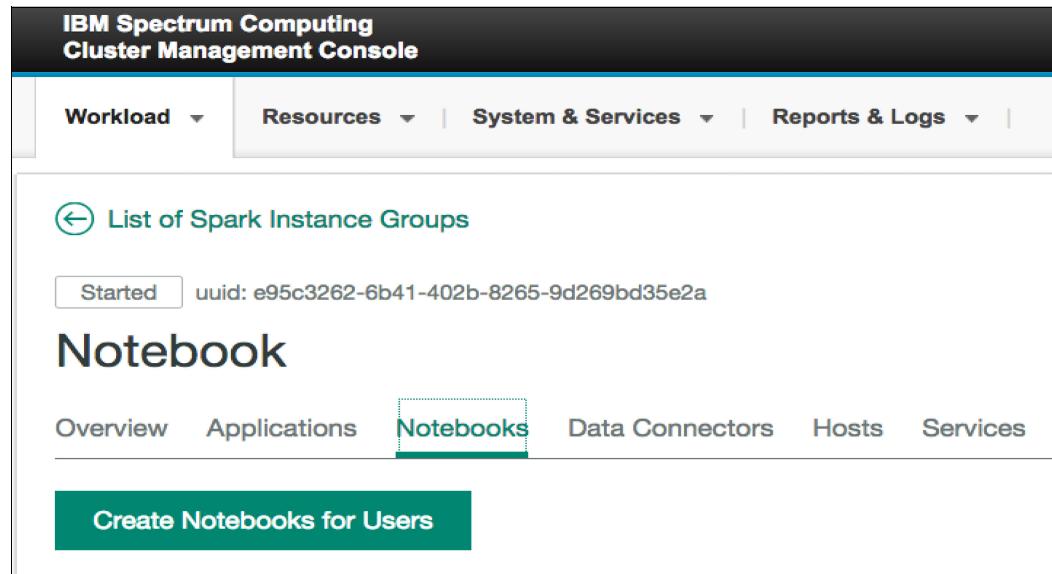


Figure B-12 The Creating Notebooks for Users menu option

2. Select the notebook and users to create and click **Create**, as shown in Figure B-13.

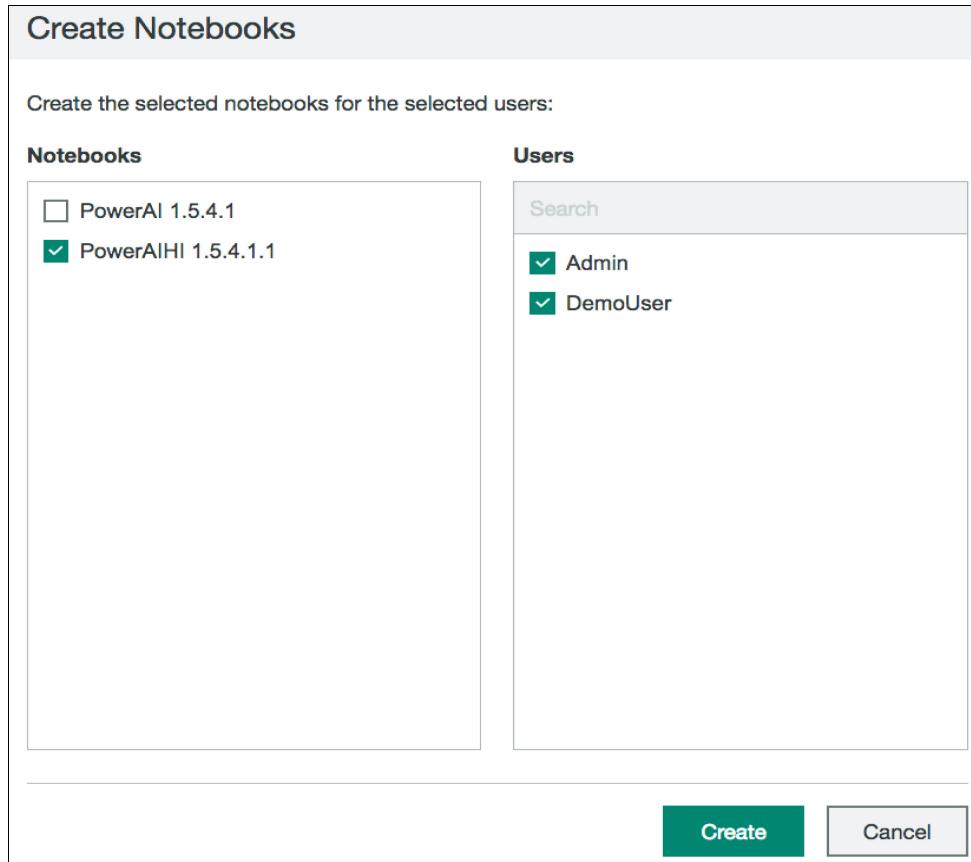


Figure B-13 Creating Notebooks for Users details

3. The notebook is created. You can access it by clicking **My Notebooks**, as shown in Figure B-14.

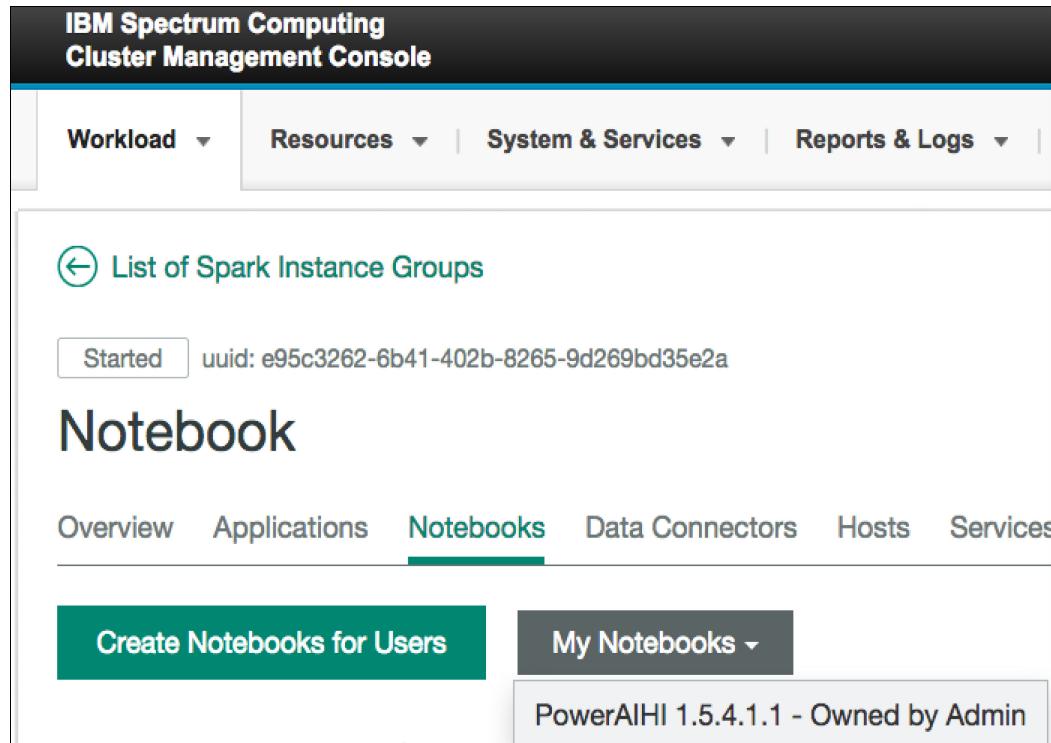


Figure B-14 My Notebooks window

4. A list of the notebook services are shown. Select the notebook in which you are interested. The notebook URL opens in a browser window.

Testing notebooks

To test a notebook, complete the following steps:

1. Log in to the notebook service, as shown in Figure B-15.

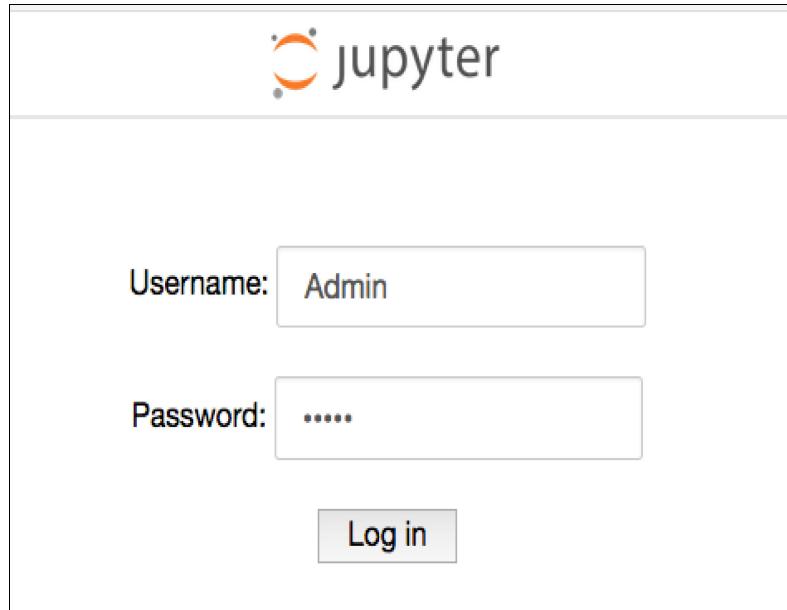


Figure B-15 Notebook server login

2. Create a Python 3 notebook by selecting **New → Python 3**, as shown in Figure B-16.

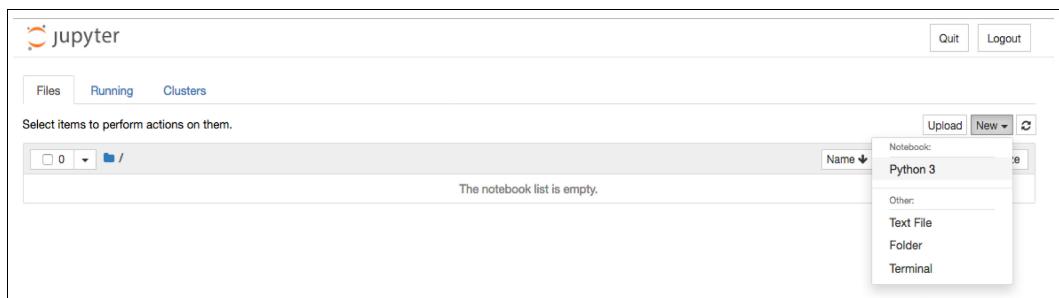
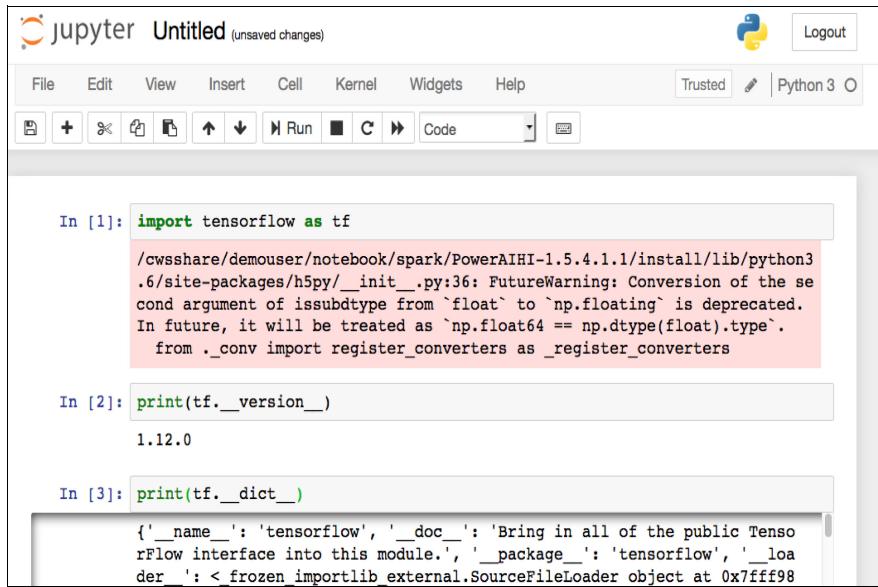


Figure B-16 Notebook server home page

A simple notebook test can output the artificial intelligence (AI) framework Python dictionary by using a code fragment similar to the following one:

```
import <framework>
print(framework.__version__)
print(framework.__dict__)
```

An example of the test output is shown in Figure B-17.



```
In [1]: import tensorflow as tf
/cwsshare/demouser/notebook/spark/PowerAIHI-1.5.4.1.1/install/lib/python3
.6/site-packages/h5py/_init__.py:36: FutureWarning: Conversion of the se
cond argument of issubdtype from `float` to `np.floating` is deprecated.
In future, it will be treated as `np.float64 == np.dtype(float).type`.
    from ._conv import register_converters as _register_converters

In [2]: print(tf.__version__)
1.12.0

In [3]: print(tf.__dict__)
{'__name__': 'tensorflow', '__doc__': 'Bring in all of the public Tenso
rFlow interface into this module.', '__package__': 'tensorflow', '__loa
der__': <frozen importlib_external.SourceFileLoader object at 0x7fff98
```

Figure B-17 Notebook test

Conclusion and additional information

This appendix covered the customization of a notebook package, adding the notebook package to IBM Watson Machine Learning Accelerator, creating a SIG that includes the notebook package, starting a notebook service, and a notebook test sample.

The IBM Redbooks GitHub website includes several sample notebooks that can be downloaded from [GitHub](#) and uploaded in to the notebook server by using the Jupyter upload button.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *Enterprise Data Warehouse Optimization with Hadoop on IBM Power Systems Servers*, REDP-5476
- ▶ *Hortonworks Data Platform with IBM Spectrum Scale: Reference Guide for Building an Integrated Solution*, SG24-5448
- ▶ *IBM PowerAI: Deep Learning Unleashed on IBM Power Systems Servers*, SG24-8409
- ▶ *IBM Power System AC922 Technical Overview and Introduction*, REDP54594
- ▶ *IBM Spectrum Archive Enterprise Edition V1.2.6 Installation and Configuration Guide*, SG24-8333

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and extra materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ Additional ingestion resources
<https://hortonworks.com/products/data-platforms/hdf/>
- ▶ Base notebook package and sample notebooks
<https://github.com/IBMRredbooks/SG24-8535-AI-and-Hortonworks-Redbook>
- ▶ The conductor-h2o-driverlessai-master.zip file
<https://github.ibm.com/kjdoyle/conductor-h2o-driverlessai>
- ▶ Enabling SSL encryption for the various Hadoop components
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/enabling-ssl-for-components.html
- ▶ Fashion MNIST example
https://www.tensorflow.org/tutorials/keras/basic_classification
- ▶ Hadoop registration service
https://content-dsxlocal.mybluemix.net/docs/content/SSAS34_current/local/hadoop.html

- ▶ H2O.ai tar.sh package
<http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/install/linux-tar-sh.html>
- ▶ H2O Driverless AI
https://content-dsxlocal.mybluemix.net/docs/content/SSAS34_current/local/h20flow.html
- ▶ H2O Driverless AI: Linux tar.sh package
<https://www.h2o.ai/download/>
- ▶ IBM Elastic Storage Server (IBM ESS)
<https://www.ibm.com/us-en/marketplace/ibm-elastic-storage-server>
- ▶ IBM Power System AC922 server
<https://www.ibm.com/id-en/marketplace/power-systems-ac922>
- ▶ Installing H2O Driverless AI
<https://ibm.box.com/s/5kpojhp87qdpcmj0hnp1bccp90etbtdh>
- ▶ Installing H2O Driverless AI as an application on IBM Watson Machine Learning Accelerator: Source file
<https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/dai/rel-1.3.1-12/ppc64le-centos7/dai-1.3.1-linux-ppc64le.sh>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

AI and Big Data on IBM Power Systems Servers

(0.2"spine)
0.17" <-> 0.473"
90<->249 pages



SG24-8435-00

ISBN 0738457515

Printed in U.S.A.

Get connected

