Jeffrey Aven

Sams Teach Yourself Hadoop in 24 Hours

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself Hadoop™ in 24 Hours

ISBN-13: 978-0-672-33852-6

ISBN-10: 0-672-33852-1

Library of Congress Control Number: 2017935714

Printed in the United States of America

1 17

Trademarks

Warning and Disclaimer

Special Sales

Editor in Chief
Greg Wiegand

Acquisitions Editor
Trina MacDonald

Development Editor
Chris Zahn

Technical Editor
Adam Shook

Managing Editor
Sandra Schroeder

Project Editor
Lori Lyons

Project Manager
Dhayanidhi

Copy Editor
Abigail Manheim

Indexer
Cheryl Lenser

Proofreader
Sathya Ravi

Editorial Assistant
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

Contents at a Glance

Table of Contents

Preface

Hadoop is synonymous with Big Data, and the two are inexorably linked together. Although there have been many books written about Hadoop before this one, many of these books have been focused on one particular area of Hadoop, or required some prior experience with Hadoop. This book is different as it explores all areas of Hadoop and the Hadoop ecosystem, as well as providing an understanding and background to the genesis of Hadoop and the big data movement.

This book is also useful if you have had some exposure to Hadoop as it explores adjacent technologies such as Spark, HBase, Cassandra, and more. The book includes many diagrams, code snippets, hands-on exercises, quizzes, and Q&As, which will help you in distilling key concepts.

I have dedicated the past several years of my career to this subject area, teaching courses and consulting to clients on analytics and big data. I have seen the emergence and maturity of the big data and open source movements, and been part of its assimilation into the enterprise. I have tried to synthesize my personal learning journey over this time into this book.

I hope this book launches or assists in your journey to becoming a big data practitioner.

How This Book Is Organized

This book starts by establishing some of the basic concepts behind Hadoop, which are covered in **Part I**, **"Getting Started with Hadoop."** I also cover deployment of Hadoop both locally and in the cloud in Part I.

Part II, **"Using Hadoop,"** is focused on the programming and data interfaces available with Hadoop, which include MapReduce, Pig, Hive, Spark, and more, as well as introductions to SQL-on-Hadoop and NoSQL using HBase.

Part III, **"Managing Hadoop,"** covers scaling, management, and administration of Hadoop and its various related technologies, including advanced configuration, securing, monitoring, and troubleshooting Hadoop.

Data Used in the Exercises

Data for the Try it Yourself exercises can be downloaded from the book's Amazon Web Services (AWS) S3 bucket. If you are not familiar with AWS, don't worry—I cover this topic in the book as well.

Conventions Used in This Book

Each hour begins with "What You'll Learn in This Hour," which provides a list of bullet points highlighting the topics covered in that hour. Each hour concludes with a "Summary" page summarizing the main points covered in the hour, as well as "Q&A" and "Quiz" sections to help you consolidate your learning from that hour.

Key topics being introduced for the first time are typically *italicized* by convention. Most hours also include programming examples in numbered code listings. Where functions, commands, classes, or objects are referred to in text, they appear in `monospace` type.

Other asides in this book include the following:

NOTE

Content not integral to the subject matter but worth noting or being aware of.

TIP

TIP Subtitle

A hint or tip relating to the current topic that could be useful.

CAUTION

Caution Subtitle

Something related to the current topic that could lead to issues if not addressed.

TRY IT YOURSELF ▼

Exercise Title

An exercise related to the current topic including a step-by-step guide and descriptions of expected outputs.

About the Author

Jeffrey Aven is a big data, open source software, and cloud computing consultant and instructor based in Melbourne, Australia. Jeff has several years' experience with Hadoop, NoSQL, Spark, and related technologies, and has been involved in key roles with several major big data implementations in Australia. Jeffrey is the author of *SAMS Teach Yourself Apache Spark* and was awarded Cloudera Hadoop Instructor of the Year for APAC in 2013.

Acknowledgments

I would like to acknowledge the team at Pearson who work hard behind the scenes on all of these projects. I would also like to acknowledge Adam Shook for his help as a technical editor for this project.

We Want to Hear from You

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: feedback@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at **informit.com/register** for convenient access to any updates, downloads, or errata that might be available for this book.

HOUR 3

Deploying Hadoop

What You'll Learn in This Hour:

- Installation platforms and prerequisites
- How to install Hadoop from Apache releases
- How to deploy Hadoop using commercial distributions
- How to deploy Hadoop in the cloud using AWS

Now that you have been introduced to Hadoop and learned about its core components, HDFS and YARN and their related processes, as well as different deployment modes for Hadoop, let's look at the different options for getting a functioning Hadoop cluster up and running. By the end of this hour you will have set up a working Hadoop cluster that we will use throughout the remainder of the book.

Installation Platforms and Prerequisites

Before you install Hadoop there are a few installation requirements, prerequisites, and recommendations of which you should be aware.

Operating System Requirements

The vast majority of Hadoop implementations are platformed on Linux hosts. This is due to a number of reasons:

- The Hadoop project, although cross-platform in principle, was originally targeted at Linux. It was several years after the initial release that a Windows-compatible distribution was introduced.
- Many of the commercial vendors only support Linux.
- Many other projects in the open source and Hadoop ecosystem have compatibility issues with non-Linux platforms.

That said there are options for installing Hadoop on Windows, should this be your platform of choice. We will use Linux for all of our exercises and examples in this book, but consult the documentation for your preferred Hadoop distribution for Windows installation and support information if required.

If you are using Linux, choose a distribution you are comfortable with. All major distributions are supported (Red Hat, Centos, Ubuntu, SLES, etc.). You can even mix distributions if appropriate; for instance, master nodes running Red Hat and slave nodes running Ubuntu.

CAUTION

Don't Use Logical Volume Manager (LVM) in Linux

If you are using Linux to deploy Hadoop nodes, master or slaves, it is strongly recommended that you not use LVM in Linux. This will restrict performance, especially on slave nodes.

Hardware Requirements

Although there are no hard and fast requirements, there are some general heuristics used in sizing instances, or hosts, appropriately for roles within a Hadoop cluster. First, you need to distinguish between master and slave node instances, and their requirements.

Master Nodes

A Hadoop cluster relies on its master nodes, which host the NameNode and ResourceManager, to operate, although you can implement high availability for each subsystem as I discussed in the last hour. Failure and failover of these components is not desired. Furthermore, the master node processes, particularly the NameNode, require a large amount of memory to operate efficiently, as you will appreciate in the next hour when we dive into the internals of HDFS. Therefore, when specifying hardware requirements the following guidelines can be used for medium to large-scale production Hadoop implementations:

- 16 or more CPU cores (preferably 32)
- 128GB or more RAM (preferably 256GB)
- RAID Hard Drive Configuration (preferably with hot-swappable drives)
- Redundant power supplies
- Bonded Gigabit Ethernet or 10Gigabit Ethernet

This is only a guide, and as technology moves on quickly, these recommendations will change as well. The bottom line is that you need *carrier class* hardware with as much CPU and memory capacity as you can get!

Slave Nodes

Slave nodes do the actual work in Hadoop, both for processing and storage so they will benefit from more CPU and memory—physical memory, not virtual memory. That said, slave nodes are designed with the expectation of failure, which is one of the reasons blocks are replicated in HDFS. Slave nodes can also be scaled out linearly. For instance, you can simply add more nodes to add more aggregate storage or processing capacity to the cluster, which you cannot do with master nodes. With this in mind, economic scalability is the objective when it comes to slave nodes. The following is a guide for slave nodes for a well-subscribed, computationally intensive Hadoop cluster; for instance, a cluster hosting machine learning and in memory processing using Spark.

- 16-32 CPU cores
- 64-512 GB of RAM
- 12-24 1-4 TB hard disks in a JBOD Configuration

NOTE

JBOD

JBOD is an acronym for *just a bunch of disks*, meaning directly attached storage that is not in a RAID configuration, where each disk operates independently of the other disks. RAID is not recommended for block storage on slave nodes as the access speed is limited by the slowest disk in the array, unlike JBOD where the average speed can be greater than that of the slowest disk. JBOD has been proven to outperform RAID 0 for block storage by 30% to 50% in benchmarks conducted at Yahoo!.

CAUTION

Storing Too Much Data on Any Slave Node May Cause Issues

As slave nodes typically host the blocks in a Hadoop filesystem, and as storage costs, particularly for JBOD configurations, are relatively inexpensive, it may be tempting to allocate excess block storage capacity to each slave node. However, as you will learn in the next hour on HDFS, you need to consider the network impact of a failed node, which will trigger re-replication of all blocks that were stored on the slave node.

Slave nodes are designed to be deployed on commodity-class hardware, and yet while they still need ample processing power in the form of CPU cores and memory, as they will be executing computational and data transformation tasks, they don’t require the same degree of fault tolerance that master nodes do.

Networking Considerations

Fully distributed Hadoop clusters are very chatty, with control messages, status updates and heartbeats, block reports, data shuffling, and block replication, and there is often heavy network utilization between nodes of the cluster. If you are deploying Hadoop on-premise, you should always deploy Hadoop clusters on private subnets with dedicated switches. If you are using multiple racks for your Hadoop cluster (you will learn more about this in **Hour 21**, "**Understanding Advanced HDFS**"), you should consider redundant core and "top of rack" switches.

Hostname resolution is essential between nodes of a Hadoop cluster, so both forward and reverse DNS lookups must work correctly between each node (master-slave and slave-slave) for Hadoop to function. Either DNS or a `hosts` files can be used for resolution. IPv6 should also be disabled on all hosts in the cluster.

Time synchronization between nodes of the cluster is essential as well, as some components, such as Kerberos, which is discussed in **Hour 22**, "**Securing Hadoop**," rely on this being the case. It is recommended you use `ntp` (Network Time Protocol) to keep clocks synchronized between all nodes.

Software Requirements

As discussed, Hadoop is almost entirely written in Java and compiled to run in a Java Runtime Environment (JRE); therefore Java is a prerequisite to installing Hadoop. Current prerequisites include:

- Java Runtime Envrionment (JRE) 1.7 or above
- Java Development Kit (JDK) 1.7 or above—required if you will be compiling Java classes such as MapReduce applications

Other ecosystem projects will have their specific prerequisites; for instance, Apache Spark requires Scala and Python as well, so you should always refer to the documentation for these specific projects.

Installing Hadoop

You have numerous options for installing Hadoop and setting up Hadoop clusters. As Hadoop is a top-level Apache Software Foundation (ASF) open source project, one method is to install directly from the Apache builds on **http://hadoop.apache.org/**. To do this you first need one or more hosts, depending upon the mode you wish to use, with appropriate hardware specifications, an appropriate operating system, and a Java runtime environment available (all of the prerequisites and considerations discussed in the previous section).

Once you have this, it is simply a matter of downloading and unpacking the desired release. There may be some additional configuration to be done afterwards, but then you simply start the relevant services (master and slave node daemons) on their designated hosts and you are up and running.

Non-Commercial Hadoop

Let's deploy a Hadoop cluster using the latest Apache release now.

TRY IT YOURSELF ▼

Installing Hadoop Using the Apache Release

In this exercise we will install a pseudo-distributed mode Hadoop cluster using the latest Hadoop release downloaded from hadoop.apache.org.

As this is a test cluster the following specifications will be used in our example:

- Red Hat Enterprise Linux 7.2 (The installation steps would be similar using other Linux distributions such as Ubuntu)
- 2 CPU cores
- 8GB RAM
- 30GB HDD
- hostname: **hadoopnode0**

1. Disable SELinux (this is known to cause issues with Hadoop):

```
$ sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' \
/etc/selinux/config
```

2. Disable IPv6 (this is also known to cause issues with Hadoop):

```
$ sudo sed -i "\$anet.ipv6.conf.all.disable_ipv6 = 1" \
/etc/sysctl.conf
$ sudo sed -i "\$anet.ipv6.conf.default.disable_ipv6 = 1" \
/etc/sysctl.conf
$ sudo sysctl -p
```

3. Reboot

4. Run the `sestatus` command to ensure SELinux is not enabled:

```
$ sestatus
```

5. Install Java. We will install the OpenJDK, which will install both a JDK and JRE:

```
$ sudo yum install java-1.7.0-openjdk-devel
```

   a. Test that Java has been successfully installed by running the following command:

   ```
   $ java -version
   ```

   If Java has been installed correctly you should see output similar to the following:

   ```
   java version "1.7.0_101"
   OpenJDK Runtime Environment (rhel-2.6.6.1.el7_2-x86_64..)
   OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)
   ```

Note that depending upon which operating system you are deploying on, you may have a version of Java and a JDK installed already. In these cases it may not be necessary to install the JDK, or you may need to set up alternatives so you do not have conflicting Java versions.

6. Locate the installation path for Java, and set the `JAVA_HOME` environment variable:

```
$ export JAVA_HOME=/usr/lib/jvm/REPLACE_WITH_YOUR_PATH/
```

7. Download Hadoop from your nearest Apache download mirror. You can obtain the link by selecting the binary option for the version of your choice at **http://hadoop.apache.org/releases.html**. We will use Hadoop version 2.7.2 for our example.

```
$ wget http://REPLACE_WITH_YOUR_MIRROR/hadoop-2.7.2.tar.gz
```

8. Unpack the Hadoop release, move it into a system directory, and set an environment variable from the Hadoop home directory:

```
$ tar -xvf hadoop-2.7.2.tar.gz
$ mv hadoop-2.7.2 hadoop
$ sudo mv hadoop/ /usr/share/
$ export HADOOP_HOME=/usr/share/hadoop
```

9. Create a directory which we will use as an alternative to the Hadoop configuration directory:

```
$ sudo mkdir -p /etc/hadoop/conf
```

10. Create a `mapred-site.xml` file (I will discuss this later) in the Hadoop configuration directory:

```
$ sudo cp $HADOOP_HOME/etc/hadoop/mapred-site.xml.template \
$HADOOP_HOME/etc/hadoop/mapred-site.xml
```

11. Add JAVA_HOME environment variable to `hadoop-env.sh` (file used to source environment variables for Hadoop processes):

```
$ sed -i "\$aexport JAVA_HOME=/REPLACE_WITH_YOUR_JDK_PATH/" \
$HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Substitute the correct path to your Java home directory as defined in Step 6.

12. Create a symbolic link between the Hadoop configuration directory and the `/etc/hadoop /conf` directory created in Step 10:

```
$ sudo ln -s $HADOOP_HOME/etc/hadoop/* \
/etc/hadoop/conf/
```

13. Create a logs directory for Hadoop:

```
$ mkdir $HADOOP_HOME/logs
```

14. Create users and groups for HDFS and YARN:

```
$ sudo groupadd hadoop
$ sudo useradd -g hadoop hdfs
$ sudo useradd -g hadoop yarn
```

15. Change the group and permissions for the Hadoop release files:

```
$ sudo chgrp -R hadoop /usr/share/hadoop
$ sudo chmod -R 777 /usr/share/hadoop
```

16. Run the built in Pi Estimator example included with the Hadoop release.

```
$ cd $HADOOP_HOME
$ sudo -u hdfs bin/hadoop jar \
share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar \
pi 16 1000
```

As we have not started any daemons or initialized HDFS, this program runs in `LocalJobRunner` mode (recall that I discussed this in **Hour 2**, “**Understanding the Hadoop Cluster Architecture**”). If this runs correctly you should see output similar to the following:

```
...
Job Finished in 2.571 seconds
Estimated value of Pi is 3.14250000000000000000
```

Now let’s configure a pseudo-distributed mode Hadoop cluster from your installation.

17. Use the `vi` editor to update the `core-site.xml` file, which contains important information about the cluster, specifically the location of the namenode:

```
$ sudo vi /etc/hadoop/conf/core-site.xml
# add the following config between the <configuration>
# and </configuration> tags:
<property>
<name>fs.defaultFS</name>
<value>hdfs://hadoopnode0:8020</value>
</property>
```

Note that the value for the `fs.defaultFS` configuration parameter needs to be set to `hdfs://HOSTNAME:8020`, where the *HOSTNAME* is the name of the NameNode host, which happens to be the localhost in this case.

18. Adapt the instructions in Step 17 to similarly update the `hdfs-site.xml` file, which contains information specific to HDFS, including the replication factor, which is set to 1 in this case as it is a pseudo-distributed mode cluster:

```
sudo vi /etc/hadoop/conf/hdfs-site.xml
# add the following config between the <configuration>
# and </configuration> tags:
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

19. Adapt the instructions in Step 17 to similarly update the `yarn-site.xml` file, which contains information specific to YARN. Importantly, this configuration file contains the address of the resourcemanager for the cluster—in this case it happens to be the localhost, as we are using pseudo-distributed mode:

```
$ sudo vi /etc/hadoop/conf/yarn-site.xml
# add the following config between the <configuration>
# and </configuration> tags:
<property>
<name>yarn.resourcemanager.hostname</name>
<value>hadoopnode0</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

20. Adapt the instructions in Step 17 to similarly update the `mapred-site.xml` file, which contains information specific to running MapReduce applications using YARN:

```
$ sudo vi /etc/hadoop/conf/mapred-site.xml
# add the following config between the <configuration>
```

```
# and </configuration> tags:
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

21. Format HDFS on the NameNode:

```
$ sudo -u hdfs bin/hdfs namenode -format
```

Enter `[Y]` to re-format if prompted.

22. Start the NameNode and DataNode (HDFS) daemons:

```
$ sudo -u hdfs sbin/hadoop-daemon.sh start namenode
$ sudo -u hdfs sbin/hadoop-daemon.sh start datanode
```

23. Start the ResourceManager and NodeManager (YARN) daemons:

```
$ sudo -u yarn sbin/yarn-daemon.sh start resourcemanager
$ sudo -u yarn sbin/yarn-daemon.sh start nodemanager
```

24. Use the `jps` command included with the Java JDK to see the Java processes that are running:

```
$ sudo jps
```

You should see output similar to the following:

```
2374 DataNode
2835 Jps
2280 NameNode
2485 ResourceManager
2737 NodeManager
```

25. Create user directories and a `tmp` directory in HDFS and set the appropriate permissions and ownership:

```
$ sudo -u hdfs bin/hadoop fs -mkdir -p /user/<your_user>
$ sudo -u hdfs bin/hadoop fs -chown <your_user>:<your_user> /user/<your_user>
$ sudo -u hdfs bin/hadoop fs -mkdir /tmp
$ sudo -u hdfs bin/hadoop fs -chmod 777 /tmp
```

26. Now run the same Pi Estimator example you ran in Step 16. This will now run in pseudo-distributed mode:

```
$ bin/hadoop jar \
share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar \
pi 16 1000
```

The output you will see in the console will be similar to that in Step 16. Open a browser and go to localhost:8088. You will see the YARN ResourceManager Web UI (which I discuss in **Hour 6, "Understanding Data Processing in Hadoop"**) (**Figure 3.1**):

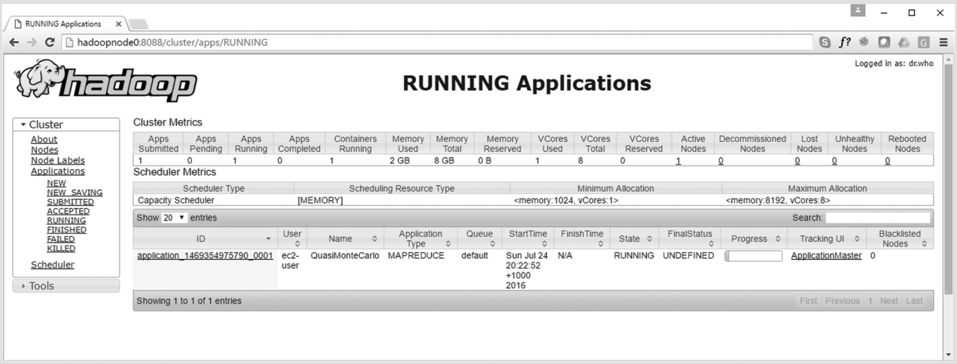

FIGURE 3.1
YARN ResourceManager Web UI.

Congratulations! You have just set up your first pseudo-distributed mode Hadoop cluster. We will use this cluster in other exercises in the book, so keep it available if you can.

Using a Commercial Hadoop Distribution

As I had discussed in **Hour 1, "Introducing Hadoop,"** the commercial Hadoop landscape is well established. With the advent of the ODPi (the Open Data Platform initiative), a once-numerous array of vendors and derivative distributions has been consolidated to a much simpler landscape which includes three primary *pure-play* Hadoop vendors:

- Cloudera
- Hortonworks
- MapR

Importantly, enterprise support agreements and subscriptions can be purchased from the various Hadoop vendors for their distributions. Each vendor also supplies a suite of management utilities to help you deploy and manage Hadoop clusters. Let's look at each of the three major pure play Hadoop vendors and their respective distributions.

Cloudera

Cloudera was the first mover in the commercial Hadoop space, establishing their first commercial release in 2008. Cloudera provides a Hadoop distribution called CDH (Cloudera Distribution of Hadoop), which includes the Hadoop core and many ecosystem projects. CDH is entirely open source.

Cloudera also provides a management utility called Cloudera Manager (which is not open source). Cloudera Manager provides a management console and framework enabling the deployment, management, and monitoring of Hadoop clusters, and which makes many administrative tasks such as setting up high availability or security much easier. The mix of open source and proprietary software is often referred to as **open core**. A screenshot showing Cloudera Manager is pictured in Figure 3.2.

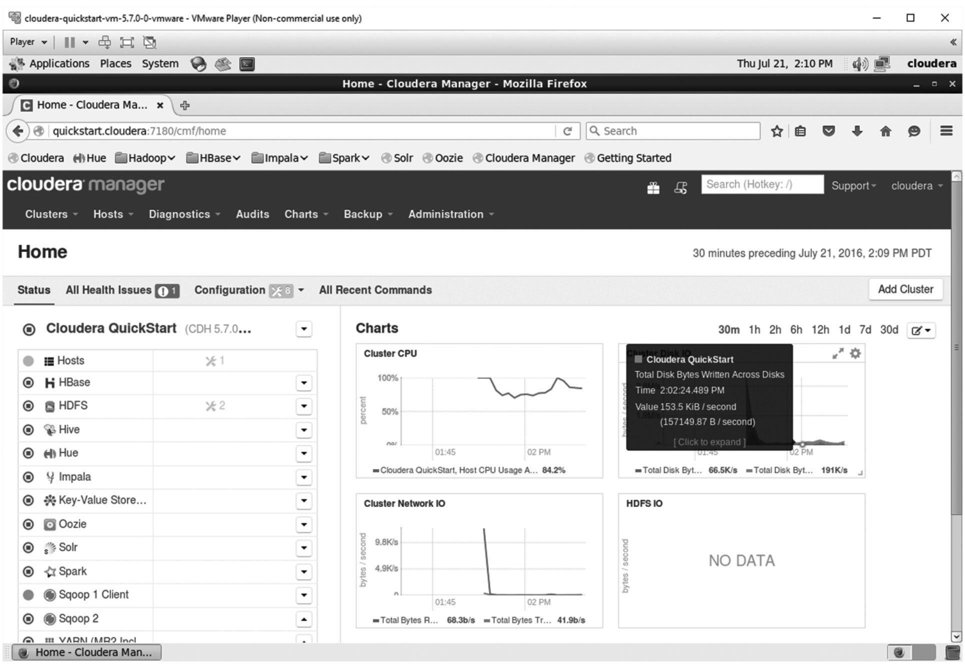

FIGURE 3.2
Cloudera Manager.

As mentioned, Cloudera Manager can be used to deploy Hadoop clusters, including master nodes, slave nodes, and ecosystem technologies. Cloudera Manager distributes installation packages for Hadoop components through a mechanism called *parcels*. As Hadoop installations are typically isolated from public networks, Cloudera Manager, which is technically not part of the cluster and will often have access to the Internet, will download parcels and distribute these to new target hosts nominated to perform roles in a Hadoop cluster or to existing hosts to upgrade components.

Deploying a fully distributed CDH cluster using Cloudera Manager would involve the following steps at a high level:

1. Install Cloudera Manager on a host that has access to other hosts targeted for roles in the cluster.
2. Specify target hosts for the cluster using Cloudera Manager.
3. Use Cloudera Manager to provision Hadoop services, including master and slave nodes.

Cloudera also provides a Quickstart virtual machine, which is a pre-configured pseudo-distributed Hadoop cluster with the entire CDH stack, including core and ecosystem components, as well as a Cloudera Manager installation. This virtual machine is available for VirtualBox, VMware, and KVM, and works with the free editions of each of the virtualization platforms. The Cloudera Quickstart VM is pictured in Figure 3.3.

FIGURE 3.3
Cloudera Quickstart VM.

The Quickstart VM is a great way to get started with the Cloudera Hadoop offering. To find out more, go to **http://www.cloudera.com/downloads.html**.

More information about Cloudera is available at **http://www.cloudera.com/**.

Hortonworks

Hortonworks provides pure open source Hadoop distribution and a founding member of the open data platform initiative (ODPi) discussed in **Hour 1**. Hortonworks delivers a distribution called HDP (Hortonworks Data Platform), which is a complete Hadoop stack including the Hadoop core and selected ecosystem components. Hortonworks uses the Apache Ambari project to provide its deployment configuration management and cluster monitoring facilities. A screenshot of Ambari is shown in Figure 3.4.

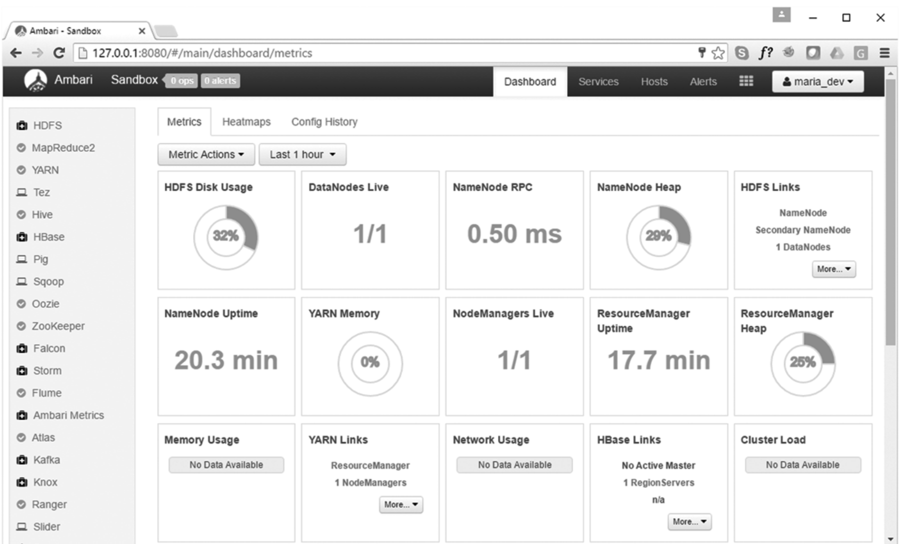

FIGURE 3.4
Ambari console.

The simplest method to deploy a Hortonworks Hadoop cluster would involve the following steps:

1. Install Ambari using the Hortonworks installer on a selected host.
2. Add hosts to the cluster using Ambari.
3. Deploy Hadoop services (such as HDFS and YARN) using Ambari.

Hortonworks provides a fully functional, pseudo-distributed HDP cluster with the complete Hortonworks application stack in a virtual machine called the Hortonworks Sandbox.
The Hortonworks Sandbox is available for VirtualBox, VMware, and KVM. The Sandbox virtual machine includes several demo applications and learning tools to use to explore Hadoop and its various projects and components. The Hortonworks Sandbox welcome screen is shown in Figure 3.5.

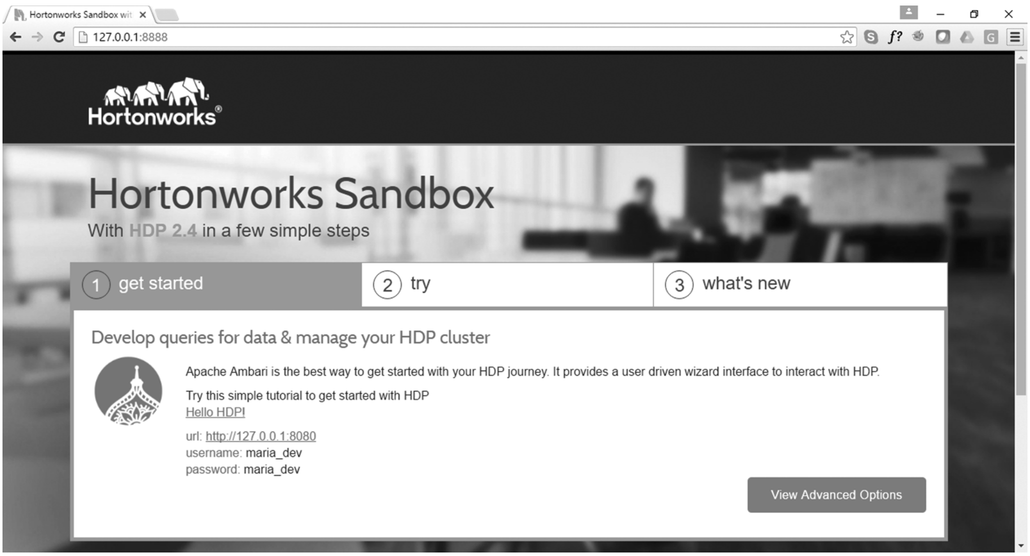

FIGURE 3.5
Hortonworks Sandbox.

You can download the Hortonworks Sandbox from **http://hortonworks.com/products/sandbox/**. More information about Hortonworks and HDP can be obtained from **http://hortonworks.com/**.

MapR

MapR delivers a "Hadoop-derived" software platform that implements an API-compatible distributed filesystem called MapRFS (the MapR distributed Filesystem). MapRFS has been designed to maximize performance and provide read-write capabilities not offered by native HDFS. MapR delivers three versions of their offering called the "Converged Data Platform." These include:

- M3 or "Converged Community Edition" (free version)
- M5 or "Converged Enterprise Edition" (supported version)
- M7 (M5 version that includes MapR's custom HBase-derived data store)

Like the other distributions, MapR has a demo offering called the "MapR Sandbox," which is available for VirtualBox or VMware. It is pictured in Figure 3.6.

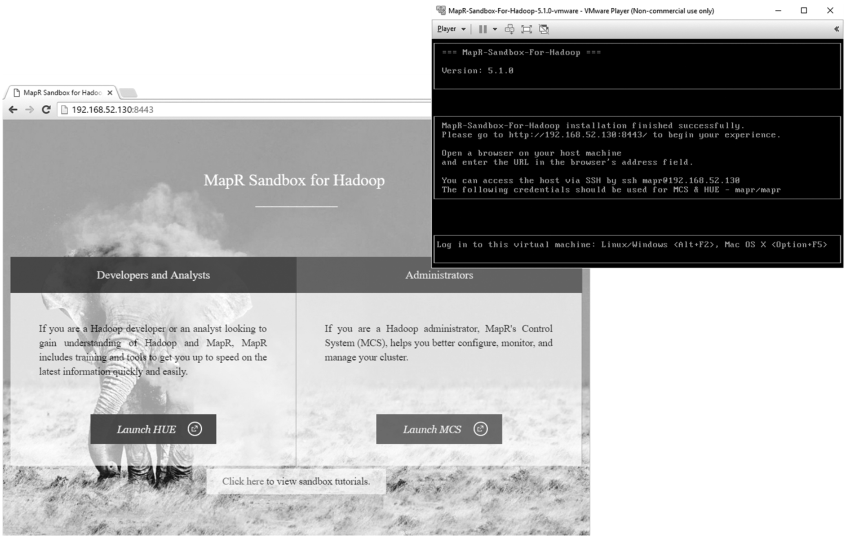

FIGURE 3.6
MapR Sandbox VM.

The MapR Sandbox can be downloaded from **https://www.mapr.com/products/mapr-sandbox-hadoop/download**.

MapR's management offering is called the MapR Control System (MCS), which offers a central console to configure, monitor and manage MapR clusters. It is shown in Figure 3.7.

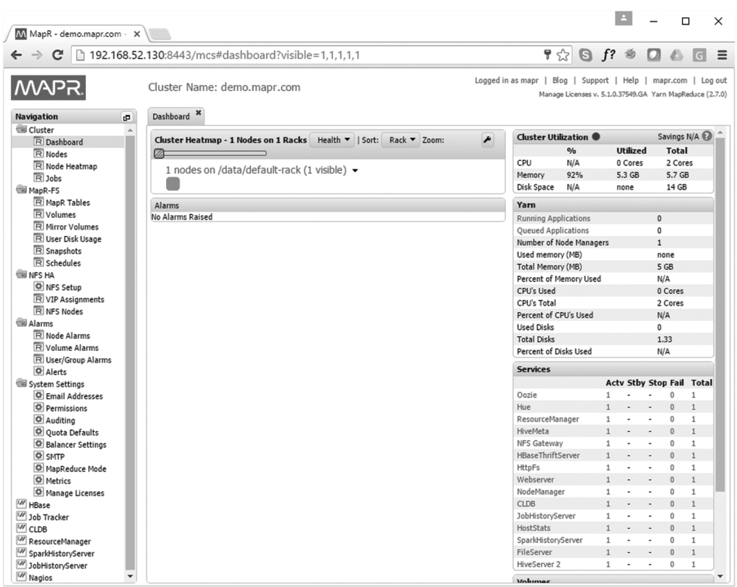

FIGURE 3.7
MapR Control System (MCS).

Much more information about MapR and the Converged Data Platform is available at **https://www.mapr.com/**.

Deploying Hadoop in the Cloud

The rise and proliferation of cloud computing and virtualization technologies has definitely been a game changer for the way organizations think about and deploy technology, and Hadoop is no exception. The availability and maturity around IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service) and SaaS (Software-as-a-Service) solutions makes deploying Hadoop in the cloud not only viable but, in some cases, desirable.

There are many public cloud variants that can be used to deploy Hadoop including Google, IBM, Rackspace, and others. Perhaps the most pervasive cloud platform to date has been AWS (Amazon Web Services), which I will use as the basis for our discussions.

Before you learn about deployment options for Hadoop in AWS, let's go through a quick primer and background on some of the key AWS components. If you are familiar with AWS, feel free to jump straight to the Try it Yourself exercise on deploying Hadoop using AWS EMR.

EC2

Elastic Compute Cloud (EC2) EC2 is Amazon's web service-enabled virtual computing platform. EC2 enables users to create virtual servers and networks in the cloud. The virtual servers are called instances. EC2 instances can be created with a variety of different instance permutations. The Instance Type property determines the number of virtual CPUs and the amount of memory and storage an EC2 instance has available to it. An example instance type is m4.large. A complete list of the different EC2 instance types available can be found at **https://aws.amazon.com/ec2/instance-types/**.

EC2 instances can be optimized for compute, memory, storage and mixed purposes and can even include GPUs (Graphics Processing Units), a popular option for machine learning and deep analytics.

There are numerous options for operating systems with EC2 instances. All of the popular Linux distributions are supported, including Red Hat, Ubuntu, and SLES, as well various Microsoft Windows options.

EC2 instances are created in security groups. Security groups govern network permissions and Access Control Lists (ACLs). Instances can also be created in a Virtual Private Cloud (VPC). A VPC is a private network, not exposed directly to the Internet. This is a popular option for organizations looking to minimize exposure of EC2 instances to the public Internet.

EC2 instances can be provisioned with various storage options, including instance storage or ephemeral storage, which are terms for volatile storage that is lost when an instance is stopped, and Elastic Block Store (EBS), which is persistent, fault-tolerant storage. There are different options with each, such as SSD (solid state) for instance storage, or provisioned IOPS with EBS.

Additionally, AWS offers Spot instances, which enable you to bid on spare Amazon EC2 computing capacity, often available at a discount compared to normal on-demand EC2 instance pricing.

EC2, as well as all other AWS services, is located in an AWS region. There are currently nine regions, which include the following:

- US East (N. Virginia)
- US West (Oregon)
- US West (N. California)
- EU (Ireland)
- EU (Frankfurt)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- South America (Sao Paulo)

S3

Simple Storage Service (S3) is Amazon's cloud-based object store. An object store manages data (such as files) as objects. These objects exist in buckets. Buckets are logical, user-created containers with properties and permissions. S3 provides APIs for users to create and manage buckets as well as to create, read, and delete objects from buckets.

The S3 bucket namespace is global, meaning that any buckets created must have a globally unique name. The AWS console or APIs will let you know if you are trying to create a bucket with a name that already exists.

S3 objects, like files in HDFS, are immutable, meaning they are write once, read many. When an S3 object is created and uploaded, an ETag is created, which is effectively a signature for the object. This can be used to ensure integrity when the object is accessed (downloaded) in the future.

There are also public buckets in S3 containing public data sets. These are datasets provided for informational or educational purposes, but they can be used for data operations such as processing with Hadoop. Public datasets, many of which are in the tens or hundreds of terabytes, are available, and topics range from historical weather data to census data, and from astronomical data to genetic data.

More information about S3 is available at **https://aws.amazon.com/s3/**.

Elastic MapReduce (EMR)

Elastic MapReduce (EMR) is Amazon's Hadoop-as-a-Service platform. EMR clusters can be provisioned using the AWS Management Console or via the AWS APIs. Options for creating EMR clusters include number of nodes, node instance types, Hadoop distribution, and additional ecosystem applications to install.

EMR clusters can read data and output results directly to and from S3. They are intended to be provisioned on demand, run a discrete workflow, a job flow, and terminate. They do have local storage, but they are not intended to run in perpetuity. You should only use this local storage for transient data.

EMR is a quick and scalable deployment method for Hadoop. More information about EMR can be found at **https://aws.amazon.com/elasticmapreduce/**.

AWS Pricing and Getting Started

AWS products, including EC2, S3, and EMR, are charged based upon usage. Each EC2 instance type within each region has an instance per hour cost associated with it. The usage costs per hour are usually relatively low and the medium- to long-term costs are quite reasonable, but the more resources you use for a longer period of time, the more you are charged.

If you have not already signed up with AWS, you're in luck! AWS has a free tier available for new accounts that enables you to use certain instance types and services for free for the first year. You can find out more at **https://aws.amazon.com/free/**. This page walks you through setting up an account with no ongoing obligations.

Once you are up and running with AWS, you can create an EMR cluster by navigating to the EMR link in the AWS console as shown in Figure 3.8.

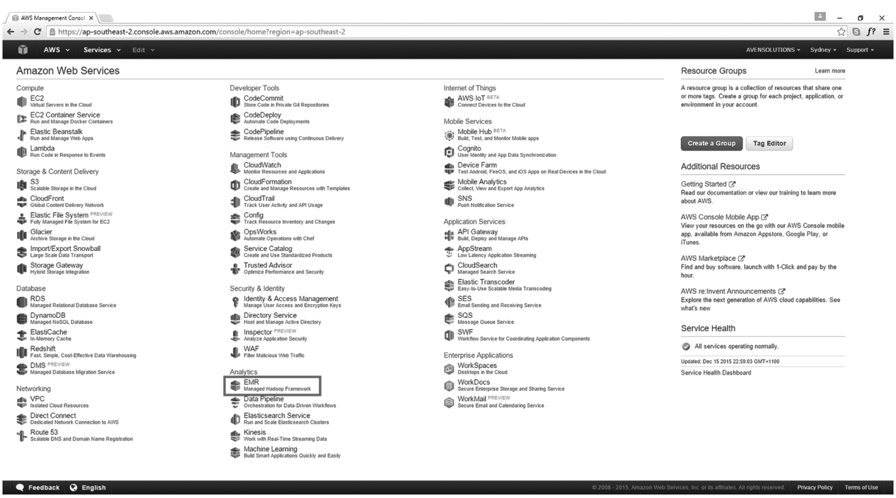

FIGURE 3.8
AWS console—EMR option.

Then click Create Cluster on the EMR welcome page as shown in Figure 3.9, and simply follow the dialog prompts.

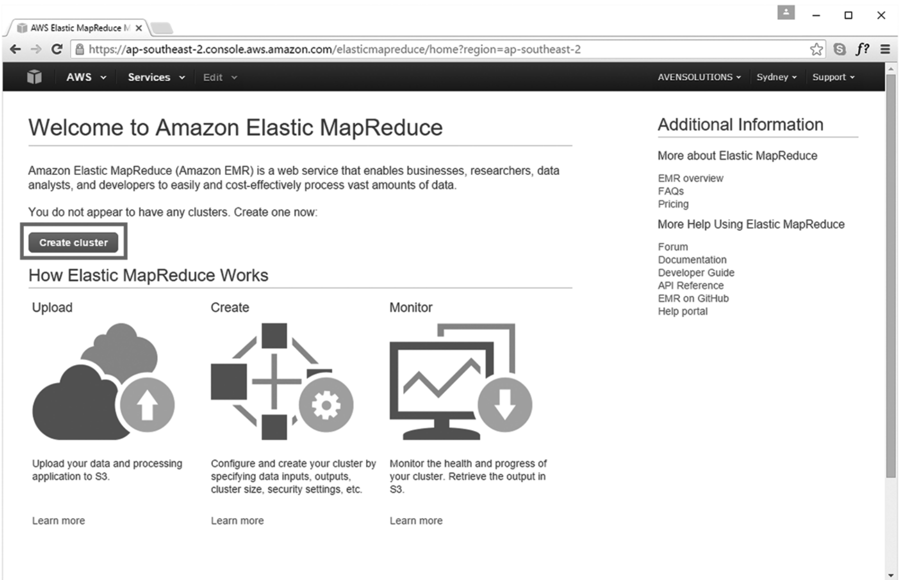

FIGURE 3.9
AWS EMR welcome screen.

You can use an EMR cluster for many of our exercises. However, be aware that leaving the cluster up and running will incur usage charges.

Summary

In this hour you learned about the various approaches to deploying a Hadoop cluster including the Apache releases, commercial distributions and cloud deployment options. Commercial distributions are often the best approach to deploying Hadoop on premise in most organizations as these distributions provide a stable, tested combination of core and ecosystem releases, as well as typically providing a suite of management capabilities useful for deploying and managing Hadoop clusters at scale.

You also learned how to provision Hadoop clusters in the cloud by using the Amazon Web Services Hadoop-as-a-Service offering—Elastic MapReduce (EMR). You are encouraged to explore all the options available to deploy Hadoop. As you progress through the book you will be performing hands-on exercises using Hadoop, so you will need to have a functional cluster. This could be one of the sandbox or quickstart commercial offerings or the Apache Hadoop cluster we set up in the Try it Yourself exercise in this hour.

Q&A

Q. Why do master nodes normally require a higher degree of fault tolerance than slave nodes?

A. Slave nodes are designed to be implemented on commodity hardware with the expectation of failure, and this enables slave nodes to scale economically. The fault tolerance and resiliency built into HDFS and YARN enables the system to recover seamlessly from a failed slave node. Master nodes are different; they are intended to be "always on." Although there are high availability implementation options for master nodes, failover is not desirable. Therefore, more local fault tolerance, such as RAID disks, dual power supplies, etc., is preferred for master nodes.

Q. What does JBOD stand for, and what is its relevance for Hadoop?

A. JBOD is an acronym for "Just a Bunch of Disks," which means spinning disks that operate independently of one another, in contrast to RAID, where disks operate as an array. JBOD is recommended for slave nodes, which are responsible for HDFS block storage. This is because the average speed of all disks on a slave node is greater than the speed of the slowest disk. By comparison, RAID read and write speeds are limited by the speed of the slowest disk in the array.

Q. What are the advantages to deploying Hadoop using a commercial distribution?

A. Commercial distributions contain a "stack" of core and ecosystem components that are tested with one another and certified for the respective distribution. The commercial vendors typically include a management application, which is very useful for managing multi-node Hadoop clusters at scale. The commercial vendors also offer enterprise support as an option as well.

Workshop

The workshop contains quiz questions to help you solidify your understanding of the material covered. Try to answer all questions before looking at the "Answers" section that follows.

Quiz

1. **True or False**: A Java Runtime Environment (JRE) is required on hosts that run Hadoop services.
2. Which AWS PaaS product is used to deploy Hadoop as a service?
   - **A.** EC2
   - **B.** EMR
   - **C.** S3
   - **D.** DynamoDB

3. Slave nodes are typically deployed on what class of hardware?

4. The open source Hadoop cluster management utility used by Hortonworks is called ______.

Answers

1. True. Hadoop services and processes are written in Java, are compiled to Java bytecode, and run in Java Virtual Machines (JVMs), and therefore require a JRE to operate.

2. B. Elastic MapReduce (EMR).

3. Commodity.

4. Ambari.

Index

A

B

C

D

L

M

N

O

T

U