O'REILLY®

Compliments of talend

Moving Hadoop to the Cloud

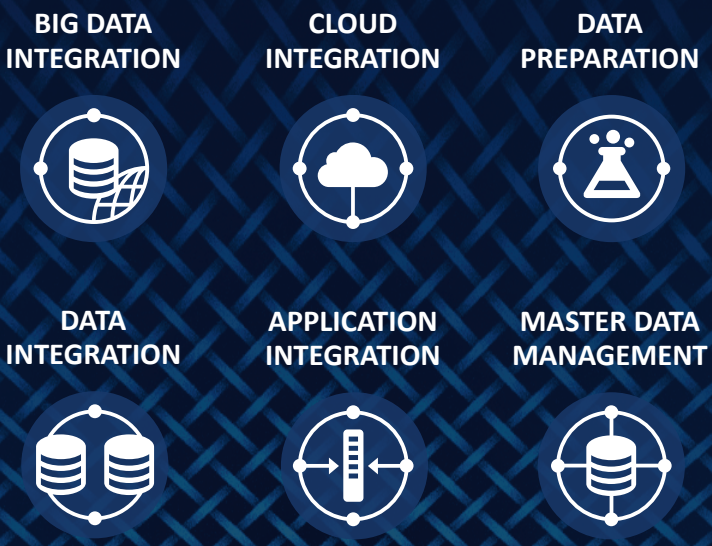
HARNESSING CLOUD FEATURES AND FLEXIBILITY FOR HADOOP CLUSTERS

Bill Havanki

talend Data Fabric

Extending Big Data Agility in the Cloud

IT leaders today are using a multi-cloud strategy to improve operational efficiency and customer centricity. The Talend Data Fabric enables enterprises to accelerate their adoption of any leading big data cloud service, quickly benefit from new innovations in the cloud, and move forward with digital transformation initiatives while future proofing current development work.



Become cloud-first now.

FREE TRIAL

Moving Hadoop to the Cloud

Harnessing Cloud Features and Flexibility for Hadoop Clusters

This Excerpt contains Chapters 1 and 5 of the book *Moving Hadoop to the Cloud*. The complete book is available at oreilly.com and through other retailers.

Bill Havanki

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Moving Hadoop to the Cloud
by Bill Havanki

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Marie Beaugureau
Copyeditor: Kim Cofer
Indexer: WordCo Indexing Services, Inc.
Cover Designer: Karen Montgomery
Production Editor: Colleen Cole
Proofreader: Christina Edwards
Interior Designer: David Futato
Illustrator: Rebecca Demarest

July 2017: First Edition

Revision History for the First Edition
2017-07-05: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491959633* for release details.

978-1-491-95963-3

[LSI]

Table of Contents

1. Why Hadoop in the Cloud? 1
What Is the Cloud? 1
What Does Hadoop in the Cloud Mean? 2
Reasons to Run Hadoop in the Cloud 3
Reasons to Not Run Hadoop in the Cloud 5
What About Security? 5
Hybrid Clouds 6
Hadoop Solutions from Cloud Providers 6
Elastic MapReduce 7
Google Cloud Dataproc 8
HDInsight 8
Hadoop-Like Services 8
A Spectrum of Choices 8

2. Storage 11
Block Storage 11
Block Storage in AWS 12
Block Storage in Google Cloud Platform 12
Block Storage in Azure 13
Object Storage 13
Buckets 14
Data Objects 15
Object Access 15
Object Storage in AWS 16
Object Storage in Google Cloud Platform 17
Object Storage in Azure 17
Cloud Relational Databases 19
Cloud Relational Databases in AWS 20

Cloud Relational Databases in Google Cloud Platform 20
Cloud Relational Databases in Azure 21
Cloud NoSQL Databases 22
Where to Start? 23

CHAPTER 1

Why Hadoop in the Cloud?

Before embarking on a new technical effort, it's important to understand what problems you're trying to solve with it. Hot new technologies come and go in the span of a few years, and it should take more than popularity to make one worth trying. The short span of computing history is littered with ideas and technologies that were once considered the future of their domains, but just didn't work out.

Apache Hadoop is a technology that has survived its initial rush of popularity by proving itself as an effective and powerful framework for tackling big data applications. It broke from many of its predecessors in the "computing at scale" space by being designed to run in a distributed fashion across large amounts of commodity hardware instead of a few, expensive computers. Many organizations have come to rely on Hadoop for dealing with the ever-increasing quantities of data that they gather. Today, it is clear what problems Hadoop can solve.

Cloud computing, on the other hand, is still a newcomer as of this writing. The term itself, "cloud," currently has a somewhat mystical connotation, often meaning different things to different people. What is the cloud made of? Where is it? What does it do? Most importantly, why would you use it?

What Is the Cloud?

A definition for what "the cloud" means for this book can be built up from a few underlying concepts and ideas.

First, a cloud is made up of computing resources, which encompasses everything from computers themselves (or *instances* in cloud terminology) to networks to storage and everything in between and around them. All that you would normally need to put together the equivalent of a server room, or even a full-blown data center, is in place and ready to be claimed, configured, and run.

The entity providing these computing resources is called a *cloud provider*. The most famous ones are companies like Amazon, Microsoft, and Google, and this book focuses on the clouds offered by these three. Their clouds can be called *public clouds* because they are available to the general public; you use computing resources that are shared, in secure ways, with many other people. In contrast, *private clouds* are run internally by (usually large) organizations.

While private clouds can work much like public ones, they are not explicitly covered in this book. You will find, though, that the basic concepts are mostly the same across cloud providers, whether public or private.

The resources that are available to you in the cloud are not just for you to use, but also to control. This means that you can start and stop instances when you want, and connect the instances together and to the outside world how you want. You can use just a small amount of resources or a huge amount, or anywhere in between. Advanced features from the provider are at your command for managing storage, performance, availability, and more. The cloud provider gives you the building blocks, but it is up to you to know how to arrange them for your needs.

Finally, you are free to use cloud provider resources for whatever you wish, within some limitations. There are quotas applied to cloud provider accounts, although these can be negotiated over time. There are also large, hard limits based on the capacity of the provider itself that you can run into. Beyond these somewhat "physical" limitations, there are legal and data security requirements, which can come from your own organization as well as the cloud provider. In general, as long as you are not abusing the cloud provider's offerings, you can do what you want. In this book, that means installing and running Hadoop clusters.

Having covered some underlying concepts, here is a definition for "the cloud" that this book builds from:

"The cloud" is a large set of computing resources made available by a cloud provider for customers to use and control for general purposes.

What Does Hadoop in the Cloud Mean?

Now that the term "cloud" has been defined, it's easy to understand what the jargony phrase "Hadoop in the cloud" means: it is running Hadoop clusters on resources offered by a cloud provider. This practice is normally compared with running Hadoop clusters on your own hardware, called *on-premises* clusters or "on-prem."

If you are already familiar with running Hadoop clusters on-prem, you will find that a lot of your knowledge and practices carry over to the cloud. After all, a cloud

instance is supposed to act almost exactly like an ordinary server you connect to remotely, with root access, and some number of CPU cores, and some amount of disk space, and so on. Once instances are networked together properly and made accessible, you can imagine that they are running in a regular data center, as opposed to a cloud provider's own data center. This illusion is intentional, so that working in a cloud provider feels familiar, and your skills still apply.

That doesn't mean there's nothing new to learn, or that the abstraction is complete. A cloud provider does not do everything for you; there are many choices and a variety of provider features to understand and consider, so that you can build not only a functioning system, but a functioning system of Hadoop clusters. Cloud providers also include features that go beyond what you can do on-prem, and Hadoop clusters can benefit from those as well.

Mature Hadoop clusters rarely run in isolation. Supporting resources around them manage data flow in and out and host specialized tools, applications backed by the clusters, and non-Hadoop servers, among other things. The supporting cast can also run in the cloud, or else dedicated networking features can help to bring them close.

Reasons to Run Hadoop in the Cloud

Many concepts have been defined so far, but the core question has not yet been answered: Why run Hadoop clusters in the cloud at all? Here are just a few reasons:

Lack of space
: Your organization may need Hadoop clusters, but you don't have anywhere to keep racks of physical servers, along with the necessary power and cooling.

Flexibility
: Without physical servers to rack up or cables to run, it is much easier to reorganize instances, or expand or contract your footprint, for changing business needs. Everything is controlled through cloud provider APIs and web consoles. Changes can be scripted and put into effect manually or even automatically and dynamically based on current conditions.

New usage patterns
: The flexibility of making changes in the cloud leads to new usage patterns that are otherwise impractical. For example, individuals can have their own instances, clusters, and even networks, without much managerial overhead. The overall budget for CPU cores in your cloud provider account can be concentrated in a set of large instances, a larger set of smaller instances, or some mixture, and can even change over time.

Speed of change
: It is much faster to launch new cloud instances or allocate new database servers than to purchase, unpack, rack, and configure physical computers. Similarly, unused resources in the cloud can be torn down swiftly, whereas unused hardware tends to linger wastefully.

Lower risk
: How much on-prem hardware should you buy? If you don't have enough, the entire business slows down. If you buy too much, you've wasted money and have idle hardware that continues to waste money. In the cloud, you can quickly and easily change how many resources you use, so there is little risk of undercommitment or overcommitment. What's more, if some resource malfunctions, you don't need to fix it; you can discard it and allocate a new one.

Focus
: An organization using a cloud provider to rent resources, instead of spending time and effort on the logistics of purchasing and maintaining its own physical hardware and networks, is free to focus on its core competencies, like using Hadoop clusters to carry out its business. This is a compelling advantage for a tech startup, for example.

Worldwide availability
: The largest cloud providers have data centers around the world, ready for you from the start. You can use resources close to where you work, or close to where your customers are, for the best performance. You can set up redundant clusters, or even entire computing environments, in multiple data centers, so that if local problems occur in one data center, you can shift to working elsewhere.

Data storage requirements
: If you have data that is required by law to be stored within specific geographic areas, you can keep it in clusters that are hosted in data centers in those areas.

Cloud provider features
: Each major cloud provider offers an ecosystem of features to support the core functions of computing, networking, and storage. To use those features most effectively, your clusters should run in the cloud provider as well.

Capacity
: Few customers tax the infrastructure of a major cloud provider. You can establish large systems in the cloud that are not nearly as easy to put together, not to mention maintain, on-prem.

Reasons to Not Run Hadoop in the Cloud

As long as you are considering why you would run Hadoop clusters in the cloud, you should also consider reasons not to. If you have any of the following reasons as goals, then running in the cloud may disappoint you:

Simplicity
: Cloud providers start you off with reasonable defaults, but then it is up to you to figure out how all of their features work and when they are appropriate. It takes a lot of experience to become proficient at picking the right types of instances and arranging networks properly.

High levels of control
: Beyond the general geographic locations of cloud provider data centers and the hardware specifications that providers reveal for their resources, it is not possible to have exacting, precise control over your cloud architecture. You cannot tell exactly where the physical devices sit, or what the devices near them are doing, or how data across them shares the same physical network.[1] When the cloud provider has internal problems that extend beyond backup and replication strategies already in place, there's not much you can do but wait.

Unique hardware needs
: You cannot have cloud providers attach specialized peripherals or dongles to their hardware for you. If your application requires resources that exceed what a cloud provider offers, you will need to host that part on-prem away from your Hadoop clusters.

Saving money
: For one thing, you are still paying for the resources you use. The hope is that the economy of scale that a cloud provider can achieve makes it more economical for you to pay to "rent" their hardware than to run your own. You will also still need a staff that understands system administration and networking to take care of your cloud infrastructure. Inefficient architectures, especially those that leave resources running idly, can cost a lot of money in storage and data transfer costs.

What About Security?

The idea of sharing resources with many other, unknown parties is sure to raise questions about whether using a public cloud provider can possibly be secure. Could other tenants somehow gain access to your instances, or snoop on the shared net-

1 An exception: some cloud providers have infrastructure dedicated to US government use where stricter controls are in place.

work infrastructure? How safe is data stashed away in cloud services? Is security a reason to avoid using public cloud providers?

There are valid arguments on both sides of this question, and the answer for you varies depending on your needs and tolerance for risk. Public cloud providers are certainly cognizant of security requirements, and as you'll see throughout this book, they use many different mechanisms to keep your resources private to you and give you control over who can see and do what. When you use a cloud provider, you gain their expertise in building and maintaining secure systems, including backup management, replication, availability, encryption support, and network management. So, it is reasonable to expect that clusters running in the cloud can be secure.

Still, there may be overriding reasons why some data simply cannot be put up into the cloud, for any reason, or why it's too risky to move data to, from, and around the cloud. In these situations, limited use of the cloud may still be possible.

Hybrid Clouds

Running Hadoop clusters in the cloud has compelling advantages, but the disadvantages may restrict you from completely abandoning an on-prem infrastructure. In a situation like that, a *hybrid cloud* architecture may be helpful. Instead of running your clusters and associated applications completely in the cloud or completely on-prem, the overall system is split between the two. Data channels are established between the cloud and on-prem worlds to connect the components needed to perform work.

Creating a hybrid cloud architecture is more challenging than running only on-prem or only in the cloud, but you are still able to benefit from some advantages of the cloud that you otherwise couldn't.

Hadoop Solutions from Cloud Providers

There are ways to take advantage of Hadoop technologies without doing the work of creating your own Hadoop clusters. Cloud providers offer prepackaged compute services that use Hadoop under the hood, but manage most of the cluster management work themselves. You simply point the services to your data and provide them with the jobs to run, and they handle the rest, delivering results back to you. You still pay for the resources used, as well as the use of the service, but save on all of the administrative work.

So, why ever roll your own clusters when these services exist? There are some good reasons:[2]

- Prepackaged services aim to cover the most common uses of Hadoop, such as individual MapReduce or Spark jobs. Their features may not be sufficient for more complex requirements, and may not offer Hadoop components that you already rely on or wish to employ.
- The services obviously only work on the cloud provider offering them. Some organizations are worried about being "locked in" to a single provider, unable to take advantage of competition between the providers.
- Useful applications that run on top of Hadoop clusters may not be compatible with a prepackaged provider service.
- It may not be possible to satisfy data security or tracking requirements with a prepackaged service, since you lack direct control over the resources.

Despite the downsides, you should investigate Hadoop-based provider solutions before rushing into running your own clusters. They can be useful and powerful, save you a lot of work, and get you running in the cloud more quickly. You can use them for prototyping work, and you may decide to keep them around for support tasks even while using your own clusters for the rest.

Here are some of the provider solutions that exist as of this writing. Keep an eye out for new ones as well.

Elastic MapReduce

Elastic MapReduce, or EMR, is Amazon Web Services' solution for managing prepackaged Hadoop clusters and running jobs on them. You can work with regular MapReduce jobs or Apache Spark jobs, and can use Apache Hive, Apache Pig, Apache HBase, and some third-party applications. Scripting hooks enable the installation of additional services. Data is typically stored in Amazon S3 or Amazon DynamoDB.

The normal mode of operation for EMR is to define the parameters for a cluster, such as its size, location, Hadoop version, and variety of services, point to where data should be read from and written to, and define steps to execute such as MapReduce or Spark jobs. EMR launches a cluster, performs the steps to generate the output data, and then tears the cluster down. However, you can leave clusters running for further use, and even resize them for greater capacity or a smaller footprint.

2 If there weren't, this book would not be very useful!

EMR provides an API so that you can automate the launching and management of Hadoop clusters.

Google Cloud Dataproc

Google Cloud Dataproc is similar to EMR, but runs within Google Cloud Platform. It offers Hadoop, Spark, Hive, and Pig, working on data that is usually stored in Google Cloud Storage. Like EMR, it supports both transient and long-running clusters, cluster resizing, and scripts for installing additional services. It can also be controlled through an API.

HDInsight

Microsoft Azure's prepackaged solution, called HDInsight, is built on top of the Hortonworks Data Platform (HDP). The service defines cluster types for Hadoop, Spark, Apache Storm, and HBase; other components like Hive and Pig are included as well. Clusters can be integrated with other tools like Microsoft R Server and Apache Solr through scripted installation and configuration. HDInsight clusters work with Azure Blob Storage and Azure Data Lake Store for reading and writing data used in jobs. You control whether clusters are torn down after their work is done or left running, and clusters can be resized. Apache Ambari is included in clusters for management through its API.

Hadoop-Like Services

The solutions just listed are explicitly based on Hadoop. Cloud providers also offer other services, based on different technologies, for managing and analyzing large amounts of data. Some offer SQL-like query capabilities similar to Hive or Apache Impala, and others offer processing pipelines like Apache Oozie. It may be possible to use those services to augment Hadoop clusters, managed either directly or through the cloud provider's own prepackaged solution, depending on where and how data is stored.

Of course, these tools share the same disadvantages as the Hadoop-based solutions in terms of moving further away from the open source world and its interoperability benefits. Since they are not based on Hadoop, there is a separate learning curve for them, and the effort could be wasted if they are ever discarded in favor of something that works on Hadoop, or on a different cloud provider, or even on-prem. Their ready availability and ease of use, however, can be attractive.

A Spectrum of Choices

It's perhaps ironic that much of this chapter describes how you can avoid running Hadoop clusters in the cloud, either by sticking with on-prem clusters (either parti-

ally or completely), by using cloud provider services that take away the management work, or by using tools that do away with Hadoop completely. There is a spectrum of choices, where at one end you work with your data at a conceptual level using high-level tools, and at the other end you build workflows, analyses, and query systems from the ground up. The breadth of this spectrum may be daunting.

However, one fact remains true: *Hadoop works everywhere*. When you focus on the core components in the Hadoop ecosystem, you have the freedom and power to work however you like, wherever you like. When you stick to the common components of Hadoop, you can carry your expertise with them to wherever your code runs and your data lives.

Cloud providers are eager for you to use their resources. They offer services to take over Hadoop cluster administration for you, but they are just as happy to let you run things yourself. Running your own clusters does not require you to forgo all of the benefits of a cloud provider, and Hadoop components that you deploy and run can still make effective use of cloud services. This book, in fact, explores how.

CHAPTER 2
Storage

Hadoop clusters are about working with data, usually lots and lots of data, often orders of magnitude larger than ever before. Cloud providers supply different ways to store that data on their vast infrastructure, to complement the compute capabilities that operate on the data and the networking facilities that move the data around. Each form of storage serves a different purpose in Hadoop architectures.

Block Storage

The most common type of storage offered by a cloud provider is the disk-like storage that comes along with each instance that you provision. This storage is usually called *block storage*, but they are almost always accessed as filesystem mounts. Each unit of block storage is called a *volume* or simply a *disk*. A unit of storage may not necessarily map to a single physical device, or even to hardware directly connected to an instance's actual host hardware.

Persistent volumes survive beyond the lifetime of the initial instances that spawned them. A persistent volume can be detached from an instance and attached to another instance, in a way similar to moving physical hard drives from computer to computer. While you wouldn't usually do that with physical drives, it is much easier to do so in a cloud provider, and it opens up new usage patterns. For example, you could maintain a volume loaded with important data or applications over a long period of time, but only attach it to an instance once in a while to do work on it.

Volumes that are limited to the lives of the instances to which they are attached are called *ephemeral* volumes. Ephemeral storage is often very fast and can be large, but it is guaranteed to be eliminated when an instance is stopped or terminated, at which time its data is permanently lost. In the context of Hadoop clusters, critical cluster-wide information like HDFS namenode data should reside on persistent storage, but

information that is normally replicated across the cluster, like HDFS data copied in as the source for a job, *can* reside on ephemeral storage.

A volume is backed up by taking a *snapshot*, which preserves its exact state at an instant in time. It is common to take snapshots of a volume repeatedly over time, and cloud providers store snapshots in an incremental fashion so that they don't take up unnecessary space. If something goes wrong with a volume, perhaps due to data corruption or a rare hardware failure, then a new volume can be created from a snapshot and attached to an instance to recover. A snapshot can also be used as the basis for a new instance image, in order to generate many new identical volumes over time.

For security, the major cloud providers all support encryption at rest for data on persistent volumes. They all also automatically replicate persistent volumes, often across data centers, to avoid data loss from hardware failures.

Block Storage in AWS

The AWS component offering block storage is called Elastic Block Storage (EBS). When you provision an instance in EC2, you select an image for its root device volume, and the image determines whether that volume is a persistent volume in EBS or an ephemeral volume in the EC2 instance store. The root device volume houses the operating system and other files from the image. The physical hardware can use either magnetic or SSD storage.

After provisioning an instance, you can attach multiple additional EBS volumes, or you can swap out the EBS root device volume of an instance with another existing one. EBS volumes are resizable, although for older instance types one must be detached before it can be resized.

Some EC2 instance types support both EBS and ephemeral volumes, while others only support EBS. Those that support ephemeral volumes do so through drives that are attached to the physical hosts for the instances. Data on those ephemeral drives survive reboots of their associated instances, but not stoppages or termination. Each instance type specifies the maximum number and size of supported ephemeral volumes.

Block Storage in Google Cloud Platform

Persistent block storage for Google Compute Engine (GCE) instances are called persistent disks. When you provision an instance in GCE, a root persistent disk is automatically allocated to house the operating system and other files from the image selected for the instance.

After provisioning an instance, you can attach multiple additional persistent disks. Each persistent disk, including the root disk, can use either magnetic or SSD storage. Persistent disks can be resized at any time.

Any instance provisioned in GCE can be augmented with local SSDs, which are drives attached to the physical hosts for the instance. These drives are ephemeral storage; while data on them survives reboots of their associated instances, it is discarded when the associated instance stops or terminates. They come in a fixed size and only a limited number may be attached to an instance. Like persistent disks, local SSDs support at-rest encryption.

RAM disks are another block storage option for GCE instances. A RAM disk is a virtual drive that occupies instance memory. These are even more ephemeral than local SSDs, as their data does not even survive instance restarts; however, they can be very fast.

Block Storage in Azure

Every Azure virtual machine is automatically granted two *virtual hard disks* or VHDs: one based on an image hosting the operating system, and a second temporary disk for storing data that can be lost at any time, like swap files. Both of these disks are persistent, but the content of the temporary disk is not preserved if the virtual machine is migrated to different hardware.

Additional persistent VHDs can serve as data disks to store data that needs to last; they are either created by mandate from the virtual machine image or can be attached later. The operating system disk and data disks are resizable, but the associated virtual machine must be stopped first.

All VHDs are actually stored as *page blobs* within the storage account associated with your Azure account, and the replication strategy for the storage account determines how widely disk contents are backed up across data centers. See "Object Storage in Azure" on page 17 for further discussion about storage accounts.

Azure File Storage is another block storage service, dedicated to serving file shares over the Server Message Block protocol. A file share works like a mounted disk, but it can be mounted across multiple virtual machines simultaneously.

Finally, while not necessarily qualifying as a block storage service, Azure Data Lake Store (ADLS) stores files of arbitrary size in a folder hierarchy. The service is designed to satisfy the requirements of a Hadoop-compatible file system, so cluster services such as Hive and YARN can work with it directly.

Object Storage

Disk-like block storage is clearly essential for supporting instances, but there is still the problem of where to keep large amounts of data that should survive beyond the lifetime of instances. You may have a compressed archive of some massive data set that will be referenced across multiple instances, or even from outside the cloud pro-

vider. You may have a backup of some important analytic results or critical logs that must be preserved. Sometimes it is possible to dedicate a block storage volume to store these big chunks of data, but there still must be at least one instance running to access it, and often it can be tricky to share that volume across multiple instances.

As an alternative, cloud providers offer *object storage*. In object storage, each chunk of data is treated as its own entity, independent of any instance. The contents of each object are opaque to the provider. Instead of accessing a data object through a filesystem mounted on a running instance, you access it through either API operations or through URLs.

Cloud providers each offer their own object storage solution, yet they all share many common features.

Buckets

Data objects reside inside containers called *buckets*. A bucket has a name and is associated with one or more regions.

Azure calls its buckets *containers*.

There are restrictions on bucket names, because a bucket's name is used as part of the URLs for accessing objects in the bucket. In general, you should avoid special characters, spaces, and other characters that can be problematic in URLs. A bucket name must be unique to your cloud provider account; the solutions for AWS and Google Cloud Platform also require them to be globally unique.

The region or regions associated with a bucket determine where in the world the objects in the bucket are stored. Hadoop clusters benefit in performance and cost by working with buckets that are in the same region as their instances. A bucket can be configured with replication to other regions, through provider-specific mechanisms, to geographically disperse data. This can be a valuable tool for expanding Hadoop architectures across regions; instead of clusters around the world all needing to reach back to a single region to access bucket contents, the bucket can be replicated to the clusters' regions for faster, cheaper local access.

A bucket does not have an internal hierarchy for object storage, but object naming can be used to create the appearance of one.

Data Objects

A data object in object storage is a single opaque unit from the cloud provider's point of view. Metadata in the form of key-value pairs can be associated with each object, to use as tags or as guides for searching for, identification of, and tracking of the data within.

The name of an object is used to locate it. As with buckets, there are restrictions on object names since they are also used in URLs. Although buckets are flat storage, objects can include forward slashes in their names to create the appearance of a directory-style hierarchy. The APIs, tools, and conventions for object storage interpret object names in a way that supports hierarchical access, often by treating common slash-delimited prefixes of object names as pseudodirectories.

An object has a *storage class*, which determines how quickly it can be accessed and, in part, its storage cost. The standard or default storage classes for cloud providers favor quick access over cost, and these classes tend to be the ones most useful for storing data Hadoop clusters will use. Other storage classes cost less but aim at access frequencies on the order of a few times each month or each year. While Hadoop clusters cannot effectively use objects in those storage classes directly due to their intentionally lower performance, they can be employed in associated archival strategies.

Most of the time, a data object is immutable. When an object needs to be updated, it is overwritten with a completely new version of it. Cloud providers are capable of storing past, versioned copies of objects so that they can be restored as necessary in the future.

AWS and Google Cloud Platform object storage services automatically version each data object as it is updated. Azure's object storage service does not, but permits taking snapshots of data objects.

Each data object can have permissions associated with it, in order to restrict who may access it. The permissions scheme folds in with each cloud provider's own authorization systems.

Object Access

There are two main methods for accessing objects in object storage. The most flexible method is through the cloud provider's own API, which provides all of the access and management operations available. An API client can create new objects, update them, and delete them. It can manage access permissions and versioning as well.

Jobs running in Hadoop clusters can use APIs to work with objects, either directly or through common libraries. Some libraries provide a filesystem-like view of buckets,

enough to allow access in many of the same ways that jobs would work with HDFS. This is a powerful capability, because it lets new clusters start their work right away by reading from durable, reliable object storage, instead of waiting to be primed by getting data copied up to their local HDFS storage. By the same token, clusters can save their final results back to object storage for safekeeping or access by other tools; once that happens, the clusters could be destroyed. Object storage access enables the use of *transient* clusters.

The other main method for object access is through URLs. Each provider's object storage service offers a REST API for working with buckets and objects; perhaps the most salient capability is simply the ability to download an object with an HTTP GET request. Because of URL access, each bucket and object must be named to be compatible with URL syntax.

For example, an object named "my-dir/my-object" in a bucket named "my-bucket" could be accessed by the following URLs, depending on the cloud provider. Each provider supports a handful of different URL formats:

- AWS: *https://s3.amazonaws.com/my-bucket/my-dir/my-object*
- Google Cloud Platform: *https://storage.cloud.google.com/my-bucket/my-dir/my-object*
- Azure: *https://my-account.blob.core.windows.net/my-bucket/my-dir/my-object*

The following sections get into more detail about the differences in the object storage services offered by major cloud providers.

Object Storage in AWS

The AWS component offering object storage is called Simple Storage Service, or S3. It stores objects in buckets that are each associated with a single region that determines where objects are stored.[1] A bucket can be configured to replicate data to a bucket in a different region for redundancy.

S3 offers four storage classes, and a bucket may hold objects in any mix of classes:

- The *standard* storage class is aimed at fast, frequent data access.
- The *standard-ia* or *standard-infrequent access* storage class carries the same durability guarantees as the standard class. It costs less to store data in this class, but each retrieval has an associated fee.

1 The choice of region also influences how data may be accessed.

- The *reduced redundancy storage* or *RRS* storage class is just as fast as the standard class but has lower durability guarantees. It costs less, and should be used for data that can be regenerated if necessary.
- The *Glacier* storage class is for nonrealtime, long-term, cheap data storage. Data must be "restored" from Glacier before it can be accessed.

The storage class for each object is selected when it is first created. Lifecycle configuration rules can be attached to buckets to automatically migrate data from more accessible to less accessible storage classes over time, so that less frequently accessed data can be stored more cheaply.

Object Storage in Google Cloud Platform

Google Cloud Storage is the service offering object storage for Google Cloud Platform. It stores objects in buckets, which are each associated with a location and a default storage class.

Google Cloud Storage offers four storage classes, and a bucket may hold objects in several classes:

- The *regional* storage class is aimed at fast, frequent data access, with storage confined to a single region determined by the bucket location.
- The *multi-regional* storage class is similar to ordinary regional storage, but data is replicated across multiple regions, determined by the bucket location. It costs somewhat more than regional storage.
- The *nearline* storage class carries the same durability guarantees as the regional and multi-regional classes. It costs less to store data in this class, but each retrieval has an associated fee.
- The *coldline* storage class is for long-term, cheap data storage. Retrieval for objects in coldline storage costs much more than for nearline storage.

The storage class for each object is selected when it is first created, or defaults to that of the bucket. Lifecycle policies can be attached to buckets to automatically migrate data from more accessible to less accessible storage classes over time, so that less frequently accessed data can be stored more inexpensively.

Object Storage in Azure

Azure has a few different forms of data storage. The service that is most pertinent to Hadoop clusters is Azure Blob Storage.

Azure Blob Storage stores objects, which are called *blobs*.[2] There are a few types of blobs:

- A *block blob* is an immutable, opaque data object.
- An *append blob* can have additional data appended to it, and can be used to receive a stream of data such as continuously generated logs.
- A *page blob* is used by Azure as the backing storage for disks.

Of the three types of blobs, the block blob is the type most useful for Hadoop clusters. Page blobs are used to store virtual machine disks, but it's not necessary to work with them directly.

Each blob is stored in a container. Each container, in turn, is associated with a *storage account* that is part of your Azure account. You can have multiple storage accounts, and each serves as a home for object storage containers and other forms of cloud storage offered by Azure. It is notable that the name for a storage account must be globally unique, but this means that container names only need to be unique within a storage account.

There are two kinds of storage accounts: general-purpose and blob. A blob storage account allows for specifying one of two access tiers for the objects stored within it:

- The *hot* access tier is aimed at fast, frequent data access. It has higher storage cost, but lower access and transaction costs.
- The *cold* access tier carries the same durability guarantees as the hot access tier. It costs less to store data in this tier, but it has higher access and transaction costs.

The storage account also determines the primary region where objects are stored, as part of the account's replication strategy. There are four replication strategies available:

- *Locally redundant storage*, or LRS, has the highest performance, but the lowest durability and is bound to a single data center in a single region.
- *Zone redundant storage*, or ZRS, goes further than LRS and spreads data out to multiple data centers, either in a single region or a pair of regions. However, it has limitations in when it can be used; in particular, a blob storage account cannot use ZRS.
- *Geo-redundant storage*, or GRS, is similar to ZRS in that it replicates data out to a secondary region.

2 The term "blob," which simply refers to a large chunk of data, has an interesting etymology.

- *Read-access geo-redundant storage*, or RA-GRS, works like GRS but also allows replicated data to be read from the secondary region.

The access tier and replication strategy for each object is determined by the storage account governing the container where the object is created. With some limitations and some costs, the access tier and replication strategy for a storage account can be changed over time.

Cloud Relational Databases

Although Hadoop breaks away from the traditional model of storing data in relational databases, some components still need their own databases to work their best. For example, the Apache Hive metastore database is crucial for mapping non-relational data stored in HDFS into a relational model that allows for SQL queries. The Apache Oozie workflow scheduling system also requires a backing database. While these components and others often can work using embedded databases, supported by software like Apache Derby, they are only production-ready when their databases are stored on full-fledged servers. By doing so, those database servers can themselves be maintained, backed up, and even shared across components, as part of an overarching enterprise architecture.

It is also helpful in some situations to simply have relational databases store some data that Hadoop clusters use or refer to. Using Hadoop does not require completely abandoning other useful technologies. Analysis jobs can refer to tables in outside databases to find metadata about the information being analyzed, or as lookup or translation mechanisms, or for numerous other helpful purposes.

Finally, applications that work with Hadoop clusters may require their own relational databases. Perhaps authentication and authorization information resides in one, or the definitions of data views reside in another.

It is perfectly reasonable to use an ordinary cloud instance as the host for a database server. It is a familiar and comfortable arrangement for database administrators, and lets you carry over existing maintenance processes into the cloud. You have direct control over upgrades, downtime, backups, and so on, and additional applications can be installed alongside the server as necessary. There are images available for popular databases to make standing up a database server instance even easier.

However, cloud providers also offer native, abstracted services for supporting database servers. With these services, instead of requesting a compute instance, you request a database server instance, specifying the type, version, size, capacity, administrative user credentials, and many other parameters. The cloud provider handles setting up the server and ensuring it is accessible to your virtual networks as you request. You do not have login access to the instance, but can access the database

server process using all the usual client tools and libraries in order to define schemas and load and query data. The provider handles backups and ensures availability.

Hadoop components can easily use cloud relational databases. The servers have hostnames and ports, and you can define the users, accesses, and schemas each component requires. From the components' points of view, the servers host remote databases like any other.

Cloud Relational Databases in AWS

The AWS component offering cloud relational databases is called Relational Database Service, or RDS. Each *database instance* hosts a single database server, and is similar in concept to an EC2 instance, having an *instance class* that determines its compute power and memory capacity. Storage capacity is managed separately, ranging from gigabytes to terabytes.

Like ordinary instances, database instances run inside an availability zone associated with a subnet inside a VPC network, and security groups govern access. Moreover, a multi-AZ database instance can continuously replicate its data to another in a separate availability zone.

RDS also automatically takes care of performing periodic database backups and minor server software upgrades. It also handles automatically failing over a multi-AZ database instance in case of the loss of the primary database instance, due to an availability zone outage or a hardware failure.

RDS supports many popular database types, including MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server. It also offers Aurora, a MySQL-compatible database type that has some additional benefits, including the ability to establish a database cluster. For database types that require licenses, such as Oracle and Microsoft SQL Server, you have the option of bringing your own license (BYOL) or establishing instances that AWS has already licensed.

Cloud Relational Databases in Google Cloud Platform

Google Cloud SQL is the primary service offering cloud relational databases for Google Cloud Platform. A Cloud SQL instance hosts a database server in a specific region and availability zone, and like an ordinary instance has a *machine type* determining the number of virtual CPUs and the size of its memory. Storage capacity is specified separately and can be set to grow automatically as needed.

Network access to Cloud SQL instances can be granted by whitelisting the IP addresses of GCE instances, but an easier method is to use a Cloud SQL proxy, which provides an encrypted tunnel to a Cloud SQL instance. Client use of a proxy works just like ordinary database access, and the proxy can be configured with a list of Cloud SQL instances to support or perform automatic discovery.

A Cloud SQL instance can be set up for high availability, which causes the creation of a second instance serving as a failover replica. If the primary instance becomes unavailable, Google Cloud SQL automatically switches to the replica, redirecting the hostname and IP address accordingly. Google Cloud SQL also handles periodic database backups.

Only MySQL is supported by Google Cloud SQL.

Google Cloud Spanner

In beta at the time of writing, Google Cloud Spanner is another, new cloud relational database service that takes advantage of Google's robust global network to natively provide high availability while enforcing strong consistency. Cloud Spanner automatically maintains replicas across availability zones in the chosen region where an instance resides. What's more, it provides ways to control where related data is located through features like parent-child table relationships and interleaved tables.

Cloud Relational Databases in Azure

The Azure SQL Database service focuses primarily on the creation and maintenance of relational databases, as opposed to the servers that host them. A database belongs to a *service tier*, which defines the guaranteed amount of compute, memory, storage, and I/O resources available for database access. Each database runs on a server; multiple databases can reside on a single server.

A database server belongs to an Azure resource group, which determines the region where the server resides. Firewall rules associated with a database server control which ranges of IP addresses are permissible for client access.

The service tier of a database can be updated over time in response to changing needs, but a more powerful approach is to use *elastic pools*. Resource usage is spread across all of the databases belonging to the pool, and administrative tasks can be conveniently performed *en masse*.

Azure SQL Database handles performing automatic backups, storing data both local to each database and in a separate data center. Long-term backups can be sent to a vault. You can also set up replication to several additional databases in other regions, any of which can be promoted to serve as a new primary database.

Azure SQL Database fully supports Microsoft SQL Server. However, newer services such as Azure Database for MySQL and Azure Database for PostgreSQL, support

those open source databases in a similar fashion, including tiered guarantees on size and performance and automatic replication.

Azure Cosmos DB

Newly introduced at the time of writing, Azure Cosmos DB is a service for managing globally distributed databases that are either relational or nonrelational, using a common data storage model beneath. The service gives you some control over how data is partitioned and automatically scales storage to achieve the desired throughput. You can also choose from one of several consistency models to guide the timeliness of data availability. Behind the scenes, Cosmos DB spreads data globally to fit your desired associations, and you can control whether to focus service interactions on specific regions or across the world.

Cloud NoSQL Databases

It is in the same spirit of offering relational database services that cloud providers also offer services that provide nonrelational or "NoSQL" databases. These services are usually pushed to the margins when working with Hadoop clusters, since Hadoop's own data storage technologies take center stage. However, as with relational databases, NoSQL databases can be useful for ancillary applications associated with the cluster.

In general, the databases hosted by NoSQL database services are highly scalable both in size and geolocation, and offer high performance. The cloud provider handles administration and availability concerns, much like they do for relational databases. Access control is managed by the cloud provider's own identity and access management systems.

The primary drawback of NoSQL database services is that they are much more specific to each cloud provider. While any well-supported relational database supports SQL, each cloud provider's NoSQL database service has a way, or set of ways, of working with data that differs from other services, sometimes even those from the same provider. This can contribute to becoming tied to a single cloud provider, which may be undesirable from a competitive point of view. Also, Hadoop components rarely have use for a NoSQL database beyond whatever storage is set up within their clusters.

Still, NoSQL database services are a part of the storage services available from cloud providers, and they can find their place in some architectures. Here is a quick rundown of what is available at the time of writing:

- AWS offers the DynamoDB database service. Tables can be accessed through the AWS console or through a variety of client-side libraries.
- Google Cloud Platform offers two NoSQL database services. Cloud Datastore works well for frequent transactions and queries and has higher durability, while Bigtable emphasizes speed and supports access through the API for Apache HBase.
- Azure's DocumentDB service, now part of Azure Cosmos DB, stores databases containing collections of documents, and can be accessed through a RESTful protocol as well as the MongoDB API. Azure Table Storage, which is associated with storage accounts, works with entity-based data.

Where to Start?

This chapter has covered four distinct forms of storage supported by cloud providers: block, object, relational, and NoSQL. Each of these is provided by one or more cloud provider services, and those are just a part of those providers' suites of services. It can be a lot to take in, especially when figuring out how Hadoop clusters fit.

The two more important services to think about are block storage and object storage. Block storage is the most crucial, since it provides the disk volumes that instances need to run. Every cluster needs block storage as an underpinning, and fortunately it is reasonably straightforward to work with. Object storage is a powerful addition to your clusters, giving you a place to keep data that survives cluster lifetimes and even supporting direct use by some Hadoop components.

Relational database services are handy for supporting the Hadoop components and secondary applications that work with clusters, but are not as important as block and object storage, especially since you can host your own database servers on ordinary instances. NoSQL database services are the least important, being unnecessary for most cluster architectures, but potentially useful in some cases.

Understanding the three core concepts of instances, networking, and storage, you are ready to jump in and create clusters in the cloud. The next part of this book begins with individual chapters for three major cloud providers, in which you will prepare instances and virtual networks necessary for a simple cluster. You should focus on your cloud provider of choice, although it is informative to see how similar tasks work in other providers, so you may want to skim the chapters for the other providers:

- If you are using AWS, continue with Chapter 6.
- If you are using Google Cloud Platform, continue with Chapter 7.

- If you are using Azure, continue with Chapter 8.

Once you have worked through your cloud provider's chapter, Chapter 9 pushes forward with Hadoop installation, configuration, and testing, which works much the same no matter what provider you use.

About the Author

Bill Havanki is a software engineer working for Cloudera, where he has contributed to Hadoop components as well as systems for deploying Hadoop clusters into public Cloud services. Prior to joining Cloudera he worked for 15 years developing software for government contracts, focusing mostly on analytic frameworks and authentication and authorization systems. He earned his B.S. in Electrical Engineering from Rutgers University and his M.S. in Computer Engineering from North Carolina State University. A New Jersey native, he currently lives near Annapolis, Maryland with his family.

Colophon

The animal on the cover of *Moving Hadoop to the Cloud* is a southern reedbuck (*Redunca arundinum*).

Southern reedbucks are typically found in southern Africa. They inhabit areas of tall grass near a source of water. The grass offers camouflage from predators such as lions, leopards, cheetahs, spotted hyenas, pythons, and crocodiles. Being herbivores, the tall grass also provides sustenance. Southern reedbucks need to drink water at least every few days, which is not typical for species in this arid region of Africa.

An elegant antelope, southern reedbucks have distinctive dark lines running down the front of their forelegs and lower hind legs. The color of their coat ranges between light- and greyish-brown and their underparts are white. Only the males bear forward-curving horns, about 35–45 cm (14–18 in) long.

The southern reedbuck is monogamous, a pair inhabits a territory that is defended by the male from other males. A single calf is born after a gestation period of around eight months and remains hidden in the dense grass for the next two months. During this period, the female does not stay with her young but instead visits it for 10 to 30 minutes each day. This antelope has an average lifespan of ten years.

The southern reedbuck makes a number of characteristic noises, including a shrill whistle through the nostrils, a clicking noise to alert others about danger, and a distinctive "popping" sound, caused by the inguinal glands, heard when the southern reedbuck jumps.

Many of the animals on O'Reilly covers are endangered; all of them are important to the world. To learn more about how you can help, go to *animals.oreilly.com*.

The cover image is from *Wood's Animate Creation*. The cover fonts are URW Typewriter and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.

WP233-EN