

O'REILLY®



**Early Release**  
**RAW & UNEDITED**

# Moving Hadoop to the Cloud

---

HARNESSING CLOUD FEATURES AND FLEXIBILITY  
FOR HADOOP CLUSTERS

Bill Havanki



---

# Moving Apache Hadoop to the Cloud

*Harnessing Cloud Features and Flexibility for  
Hadoop Clusters*

*Bill Havanki*

Beijing • Boston • Farnham • Sebastopol • Tokyo

**O'REILLY®**

# Deploying Apache Hadoop in the Cloud

by Bill Havanki

Copyright © 2016 Bill Havanki Jr.. All rights reserved.

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://safaribooksonline.com> ). For more information, contact our corporate/institutional sales department: 800-998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com) .

**Editor:** Marie Beaugureau

**Production Editor:** FILL IN PRODUCTION EDITOR

**Copyeditor:** FILL IN COPYEDITOR

**Proofreader:** FILL IN PROOFREADER

**Indexer:** FILL IN INDEXER

**Interior Designer:** David Futato

**Cover Designer:** Karen Montgomery

**Illustrator:** Rebecca Demarest

November 2016: First Edition

## Revision History for the First Edition

2016-11-14: First Early Release

2016-12-21: Second Early Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781491959619> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. Deploying Apache Hadoop in the Cloud, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the author(s) have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author(s) disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-95961-9

[FILL IN]

---

# Table of Contents

<b>1. Why Hadoop in the Cloud?.....</b>	<b>5</b>
What Is the Cloud?	5
What Does Hadoop in the Cloud Mean?	6
Reasons to Run Hadoop in the Cloud	7
Reasons to Not Run Hadoop in the Cloud	8
What about Security?	9
Hybrid Clouds	9
Hadoop Solutions from Cloud Providers	10
Elastic MapReduce	11
Google Cloud Dataproc	11
HDInsight	11
Hadoop-Like Services	11
A Spectrum of Choices	12
Getting Started	12
<b>2. Instances.....</b>	<b>13</b>
Instance Types	14
Regions and Availability Zones	15
Instance Control	16
Temporary Instances	17
Spot Instances	18
Preemptible Instances	18
Images	19
No Instance Is an Island	19
<b>3. Networking and Security.....</b>	<b>21</b>
Virtual Networks	21
Private DNS	24

Public IP Addresses and DNS	24
Virtual Networks and Regions	25
Routing	26
Routing in AWS	28
Routing in GCP	29
Routing in Azure	30
Network Security Rules	30
Inbound vs. Outbound	30
Allow vs. Deny	31
Network Security Rules in AWS	31
Network Security Rules in GCP	34
Network Security Rules in Azure	36
Putting Networking and Security Together	37
What about the Data?	38
<b>4. Setting up in AWS. ....</b>	<b>39</b>
Prerequisites	39
Allocating Instances	41
Generating a Key Pair	41
Launching Instances	42
Securing the Instances	48
Next Steps	48
<b>5. Standing up a Cluster. ....</b>	<b>49</b>
The JDK	49
Hadoop Accounts	50
Passwordless SSH	50
Hadoop Installation	51
HDFS and YARN Configuration	52
The Environment	52
XML Configuration Files	54
Finishing up Configuration	56
Startup	56
SSH Tunneling	56
Running a Test Job	57
What If the Job Hangs?	58
Running Basic Data Loading and Analysis	58
Wikipedia Exports	59
Analyzing a Small Export	59
Go Bigger	64

# 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 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 which 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? 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 databases 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 their 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<sup>1</sup>. 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 staff who understand 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 network 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

---

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

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.

??? explores the pattern of hybrid clouds, including some examples for when they are appropriate or even necessary. 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 pre-packaged 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<sup>2</sup>.

- Pre-packaged 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 pre-packaged provider service.
- It may not be possible to satisfy data security or tracking requirements with a pre-packaged 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.

---

<sup>2</sup> If there weren't, this book would not be very useful!

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 pre-packaged 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.

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 pre-packaged 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 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 listed above 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 pre-packaged 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 partially 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.

## Getting Started

Have you figured out why you want to run Hadoop in the cloud? Ready to get started?

If you already know which cloud provider you'll use, skip ahead to ??? for a primer on the major concepts of cloud instances, networking, and storage. Otherwise, continue to the next chapter for an overview of the major cloud providers so that you can understand the landscape.

# Instances

The core of a cloud provider’s offerings is the ability to provision instances. An *instance* is similar in concept to a virtual machine, which is an emulation of a particular computer system<sup>1</sup>. While historically virtual machines were thought of as running on a specific piece of hardware - perhaps a server in a company’s machine room - a cloud instance is thought of as running somewhere unspecified within the cloud provider’s vast infrastructure. Precisely where and how an instance is provisioned is often not revealed to you as a customer, although you do have some coarse-grained control (see “Regions and Availability Zones” on page 15 for one example). All that matters to you is that you ask for an instance, and the cloud provider brings it to life.



Azure uses the term “virtual machine” to refer to their instances.

The features of an instance, beyond its precise physical location, are up to you. You can choose from a variety of combinations of CPU power and memory, any of several operating systems, and different storage options, just to start with. Once the instance has been provisioned, you can access it over SSH through the networking capabilities of the cloud provider. From that point on, it acts and feels just like a real, physical machine. You have root access, so you can install applications, perform additional upgrades, start web servers or Hadoop daemons, and so on.

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Virtual\\_machine](https://en.wikipedia.org/wiki/Virtual_machine)

Your cloud provider account has limits that affect the number and size of instances you can provision. Those limits, along with the cost of running instances, influence how you design your cloud architecture. Given those limits, often the first step in the design process is determining which instance types to use.

## Instance Types

Cloud providers offer instances in a multitude of combinations of features. These combinations usually differentiate themselves in terms of compute power, memory, storage capacity, and performance. To make selecting a combination easier, and also to enable the cloud provider to fit all its customers' instances together efficiently on its infrastructure, a cloud provider offers a set of fixed feature combinations, like a menu. Each of these *instance types* is defined for a purpose: heavy compute capability, vast storage, economy, or simply general-purpose use.



Azure refers to instance types as “instance sizes”.

While you are free<sup>2</sup> to always select a top-of-the-line instance type with lots of everything, it will be among the most expensive of your options, and most of the time you will not need all of that power all the time. It is much more efficient, and cheaper, to pick instance types that focus on what you need the instances for. Since a cloud provider makes it easy to provision new instances, you have the option of allocating more, cheaper instances, instead of staying with a smaller number of ultra-powerful ones.

Cluster technologies like Hadoop can flourish in this sort of environment. As your needs change over time, you can scale up or scale down the number of daemons you run for different components without worrying about the physical hardware. Moreover, Hadoop components do not require top-of-the-line instance specifications to work; they can perform well on mid-range instance types. Still, you should pick instance types that focus on the features that the components need.

Advice for selecting instance types for Hadoop clusters is provided in ????. In short, roles are defined for instances in a cluster, and then the needs for each role are discussed.

---

<sup>2</sup> During the free trial period for cloud providers, you are usually restricted to only basic instance types with relatively meager specifications. Once you upgrade to a regular account, or after some amount of time, your options open up.



Cloud providers offer many different instance types, so it can be tricky to decide on which ones to use. It is somewhat easier, though, to choose where the instances will run.

## Regions and Availability Zones

An advantage of using a cloud provider is geographic distribution. A major cloud provider's infrastructure spans timezones and continents, running 24 hours a day. A data center can be constructed anywhere that has space, electricity, network connectivity, and people to maintain it. This enables you to set up your own systems in a distributed fashion, reaping all of the associated availability and performance benefits.

A cloud provider infrastructure is exposed as a set of divided, geographic areas called *regions*. For example, a cloud provider could define three regions in the continental United States (say, west, central, and east) along with two more in Europe and two in Southeast Asia. The number and coverage areas of cloud provider regions changes over time. A busy geographic area may be covered by multiple regions.

When you provision new instances, you select the region where they live. You may decide to pick the region you work in, so that network performance between you and the instance is quickest. Or, you may be setting up a failover system and decide to choose a region farther away. If you know that most of the traffic to your instance will be from customers, you may choose a region close to them. You may pick a region to satisfy legal requirements that apply to where your data may be stored.

Regions are one of the key factors in designing a large Hadoop cluster. While spreading a cluster far and wide across the world sounds attractive for availability, it takes much longer, and will cost more, for instances in different regions to communicate with each other. So, it's important to plan carefully to minimize cross-region communication while still getting the availability you need.

To help provide both availability and performance, a cloud provider defines one or more *availability zones* (AZ) within a region. Availability zones are themselves independent within a region, but they have faster interconnections. This lets you set up a distributed architecture that sits within a single region and has good performance, yet has some measure of availability. While it is rare that an entire availability zone fails, it is far rarer for an entire region to fail.



Azure does not support the concept of availability zones, but instead lets you define *availability sets*. An availability set contains instances which are spread out among multiple *fault domains*, each of which has separate power and network connections and therefore are housed in different hosts and racks<sup>3</sup>. So, rather than managing availability zones yourself for each individual instance in your architecture, you can group them into availability sets based on their roles.

Communication cost between instances in different availability zones generally costs more than between instances in the same availability zone, and this is one factor that will influence your cluster architecture. In general, you would not spread a Hadoop cluster across availability zones except to achieve high availability. ??? is dedicated to exploring high availability, and discusses the pros and cons of availability zones as a factor.

As you've seen, you have control over the specifications of an instance through selecting an instance type and some amount of control over where an instance runs. It's time to discuss how you can control an instance's existence once it has been started.

## Instance Control

Just like a real machine, even when no one is connected to it, an instance still “exists” and keeps running. This makes instances perfectly suited for hosting daemons like those across the Hadoop ecosystem. The cloud provider monitors your instances, and you can set up alerts to be notified if any of your instances become unreachable or, in rare cases, die out.

If you know that you will not be using an instance for a while, you can *stop* it. Stopping an instance works like shutting off a physical machine; processes running on it are terminated normally and the operating system halts. A stopped instance is unreachable until it is started again, which works like turning that physical machine back on.

Why stop an instance instead of just leaving it running? One important reason is that your cloud provider charges your account much less, or not at all, for an instance while it is stopped, so it is economical to stop instances you aren't using. In addition, some operations on an instance, like changing its attached storage, can only be performed when the instance is stopped.

---

<sup>3</sup> Fault domains may still reside in the same data center.



Azure continues to charge for instances that are stopped. You must also “deallocate” instances to suspend charging for instances. This is because Azure retains the compute and memory resources for your instances even when they are stopped.

Once an instance has served its purpose, it can be *terminated* or deleted. A terminated instance cannot be started up again; you need to provision a new instance to replace it. Once an instance is terminated, everything that was on its disks is lost, unless you have backed it up, either on your own or by using other cloud provider capabilities (see ???). Cloud providers stop charging for instances when they are terminated.



Terminating an instance causes a cloud provider to reclaim most or all of the resources associated with that instance, but stopping may also cause the provider to reclaim some resources, such as ephemeral storage (see ???) or public IP addresses (see “[Virtual Networks](#)” on page 21). Check your provider’s documentation for complete information. It may be the case that even stopping an instance participating in a Hadoop cluster will render it unusable when it is started again.

## Temporary Instances

By default, once you allocate an instance, it is yours to control until you terminate it yourself; it will not be taken away from you unless some rare problem occurs at the cloud provider, and even then you usually receive a warning and some lead time to react.

Under some circumstances, however, you may decide to use a *temporary instance* which can disappear after some time. While this seems like a bad idea in the general case, temporary instances can be useful for surging your capacity for a few hours, or for running some process that won’t take long.

Still, though, why not just use ordinary instances all the time? The main reason is that cloud providers charge significantly less for temporary instances than for ordinary ones. Cloud providers almost always have excess capacity going unused, so temporary instances are a way to earn revenue on it, even at a discount, until the capacity can be allocated to ordinary provisioned instances.

In order to use temporary instances effectively, you must have automated means of bootstrapping them and pressing them into service. If you spend too much time getting them set up, they may disappear before you get a chance to do anything with them. Nevertheless, if that does happen, it’s straightforward to rerun automation and try again with another temporary instance. This mode of retrying on temporary

instances is not suitable for critical pieces of your system, but can save money elsewhere.

While it makes good sense to not under-utilize temporary instances, it makes good sense as well not to over-utilize them, especially for the sake of saving money. Temporary instances *will* disappear on you, sometimes when you least expect it or want them to, and a quiet spell of weeks where temporary instances have been reliable and lingered for plenty of time can end surprisingly suddenly. So, use them, but use them wisely.



Azure does not offer temporary instances at this time.

## Spot Instances

AWS calls their temporary instances *spot instances*. There is a market for spot instances within AWS, driven by the price that customers are willing to pay for them. When demand for spot instances is low, the price is low; when demand goes up, so does the price. When you request spot instances, you select a price that you are willing to pay, anywhere from the current market price up to the fixed rate for ordinary instances.

The spot market determines not only the price for spot instances, but also how long they last. Once the spot price rises above your chosen price, your spot instances are reclaimed so that their resources can be used by those who bid higher (or for ordinary instances). So, you can choose a higher initial price for a higher probability of keeping your spot instances longer, but you may end up paying “too much” if the market price remains low.

Spot instances are particularly prone to overuse, since there is no pre-determined time after which they will disappear. They can persist for days or weeks, and lull users into a false sense of stability. Don’t be fooled; always treat them as if they could disappear at any moment.

## Preemptible Instances

GCP calls their temporary instances *preemptible instances*. Unlike AWS, there is no market determining the price for an preemptible instance; there is a single offered price which is lower than a standard instance.

While market conditions in AWS determine when your spot instances are reclaimed, preemptible instances are *guaranteed* to be reclaimed within a day (24 hours), if not sooner. This does much to reduce the temptation to over-rely on them and promotes

the practices of automating their configuration and keeping your cluster resilient to loss of instances.

## Images

Besides its instance type, location, and lifecycle, another key feature of an instance is what it's actually running: its operating system type and version, the software packages that are available, and applications that are installed. These considerations are all bundled up into *images*. An image can be thought of as the plan for a virtual drive that your instance runs from. Conceptually it is just like a virtual machine image file: an encapsulated file system with a specific operating system and other software installed and ready to go.

When you provision an instance, you select the image that it should start from. Larger cloud providers can support hundreds or thousands of different images, some that they fashion themselves, but many more that are created by operating system vendors, vendors of other software, and in some cases even individuals. Cloud providers will propose a small set of stock images to help you get started, but you have the freedom to use any image you like.

Most images are free to use, especially those that bundle free operating systems. However, there are some that cost money to use, either as a fixed additional cost or a continual surcharge to running them. In addition, some images may host unlicensed, freshly installed (at the time the image was created) software that will prompt you for license details before your instances will start to fully function. Hadoop can be used on images that include free operating systems like Ubuntu and CentOS, or on those that require licenses, such as Red Hat Enterprise Linux.

One of the most important things you can do to make using Hadoop on a cloud provider easier is to create your own images. Instead of using a basic image for every instance and installing Hadoop components on them every time, even in an automated fashion, you can instead create a custom image with the components installed where you need them. ??? goes into detail about this process.

## No Instance Is an Island

Instances provisioned by a cloud provider aren't of much use if no one can reach them. The next chapter discusses how instances are connected into a network so that you can reach them and they can reach other and other systems outside the cloud provider.



---

# Networking and Security

An essential aspect of working with instances is configuring their network connectivity. While cloud providers start customers off with a basic, working network configuration, it's important to understand the ways to construct different network topologies, so that your clusters can communicate effectively internally, and back and forth with your own systems and with the outside world.

Network topology is of primary importance when setting up Hadoop clusters. Worker daemons like datanodes and node managers must be able to work with namenodes and resource managers, and clients must understand where to send jobs to run and where cluster data resides. You will likely spend more time designing and maintaining the network architecture of your clusters than the instances and images that serve as their building blocks.

Security considerations are intertwined with network design. Once network connections are made, you need to determine the rules by which they are used. Which parts of the network can talk to which other parts of the network? What can reach out to the internet? What can reach *in* from the internet? What ports should be exposed, and to whom?

This chapter covers a wide range of topics, and is more of an introduction to cloud networks and security than an application of them to Hadoop, although there are some pointers. ??? goes into much more detail about designing a network and security rules for a Hadoop cluster.

## Virtual Networks

Cloud providers establish *virtual networks* as the top-level containers where instances live. Each virtual network is separate from other virtual networks, and instances

within a virtual network always communicate with each other more directly than with instances in other virtual networks or outside of the cloud provider.

A virtual network is just a basic, coarse-grained concept. To enable finer control of network topology, each virtual network is divided up into subnetworks or *subnets*. A subnet is not just a lower-level instance container; it also covers a range of private IP addresses. There are normally several subnets within a virtual network, each with a distinct range of *private* IP addresses.



**RFC 1918** establishes three ranges of private IP addresses. Cloud providers use these ranges to define subnets. Any of these blocks, including just portions of them, can be used for a subnet.

- 10.0.0.0 - 10.255.255.255 (CIDR 10.0.0.0/8)
- 172.16.0.0 - 172.31.255.255 (CIDR 172.16.0.0/12)
- 192.168.0.0 - 192.168.255.255 (CIDR 192.168.0.0/16)



Amazon Web Services calls its virtual networks Virtual Private Clouds or VPCs. Each VPC has a private IP address range, and the address range for each subnet within a VPC must be a subset of the VPC's range. The subnet ranges do not have to completely cover the VPC range.

For example, a single virtual network could be designed to cover the entire 16-bit private IP address block of 192.168.0.0/16. One way to divide the network is into four subnets, each covering a distinct quarter of the block: 192.168.0.0/18, 192.168.64.0/18, 192.168.128.0/18, and 192.168.192.0/18.



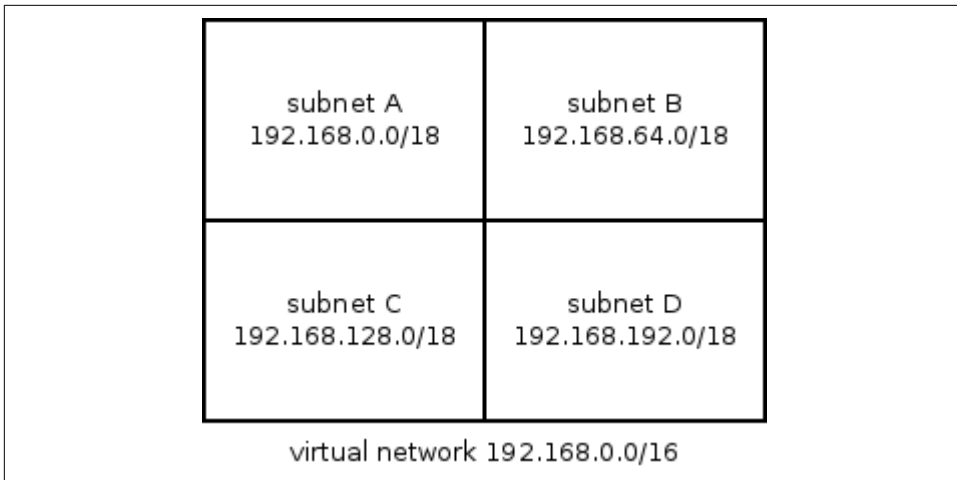


Figure 3-1. A virtual network with four subnets.

After a virtual network is established, subnets must be created within it as homes for instances that will reside in the virtual network. Sometimes the cloud provider establishes one or more default subnets, and sometimes it is up to you to define them. The size of the private IP range of a subnet dictates its capacity for instances: for example, a range like 192.168.123.0/28 only supports sixteen instances, while a range like 172.16.0.0/16 supports thousands. Instances that reside in the same subnet can communicate more quickly and easily than those in separate subnets, so sizing subnets appropriately is important for designing efficient clusters.

When you provision an instance in a cloud provider, you choose its subnet. The cloud provider assigns the instance an IP address from the remaining unused addresses in the subnet's range, and that IP address sticks with the instance until it is terminated.



The private IP address for a Azure virtual machine can either be static or dynamic. A dynamic private IP address, which is the default, is dissociated from the virtual machine even when it is stopped, while a static private IP remains across stops, until termination. In order to avoid needing to reconfigure your Hadoop cluster after virtual machines are stopped and started, you will want to use static addresses.

Most of the time, a Hadoop cluster should reside within a single subnet, itself within one virtual network. Not only is this arrangement simplest, it is the least expensive and has the best performance. ??? explores other arrangements in terms of establishing high availability.

## Private DNS

When an instance is provisioned inside a subnet, it is assigned not only a private IP address, but also a *private* DNS hostname. The hostname is automatically generated for you and registered with the cloud provider's internal DNS infrastructure. It may simply be a form of the public IP address or some random string, and thus have no meaning. The cloud provider also automatically configures each instance's network settings so that processes running on it can resolve private DNS hostnames successfully, both its own and those of others in the virtual network.

A private DNS hostname can be resolved to a private IP address *only* by instances within the virtual network of the instance it is assigned to. Other instances, including those in other virtual networks of the same cloud provider, must use a public DNS hostname or public IP address, if those are assigned at all.



In Azure, two virtual networks can be *peered*, so that instances in them can communicate using private IP addresses. The two networks must have private IP address ranges that do not overlap.

In practice, private DNS hostnames have limited usefulness in working with a Hadoop cluster; the private IP addresses work just as well, and are often shorter and therefore easier to work with. Given all the other things to think about when managing virtual networks and how instances are deployed within them, you may find that private DNS can essentially be ignored.

## Public IP Addresses and DNS

While an instance is always assigned a private IP address, it may also be assigned a public IP address. The public IP address is not part of the instance's subnet's IP range, but is instead assigned from the block of IP addresses administered by the cloud provider. An instance with a public IP address is therefore addressable from outside the virtual network and, in particular, from the internet.

While having a public IP address is prerequisite for an instance to have connectivity outside the virtual network, it does not mean that the instance *can* be reached, or itself reach out from the virtual network. That depends on the security rules that govern the instance and routing rules that apply to the subnet.

A cloud provider may also assign a public DNS hostname to an instance with a public IP address. The typical public DNS hostname is under the domain of the cloud provider and, like a private DNS hostname, often has no real meaning. Still, the cloud provider does establish resolution of the public DNS hostname to the public IP address for external clients, so it is useable.

If you have a DNS domain that you want to use to assign public DNS hostnames to your instances, you can use the cloud provider's public DNS component to manage assignments (AWS Route 53, Google Cloud DNS, Azure DNS). In the context of configuring and using Hadoop clusters, however, it's almost always sufficient to work with private DNS hostnames for instances. Save public DNS hostnames for those few gateway instances<sup>1</sup> that host public-facing interfaces to your system.



Without a public IP address or public DNS hostname, an instance is not reachable from the internet. It is therefore much more difficult to accidentally expose such an instance through, for example, overly permissive security settings.

The private and public addresses for instances in a virtual network provide a logical means for finding where instances are. It is not as obvious how to understand where a virtual network and subnets within it are located.

## Virtual Networks and Regions

The location of a subnet in a virtual network, or of an entire virtual network, is determined in different ways, depending on the cloud provider.

A subnet in AWS and GCP, besides determining the private IP address for an instance, also determines the region where the instance resides. In AWS, each subnet that you define is assigned to an availability zone, so instances in the subnet run in that availability zone, in the zone's region. In Google Cloud, the arrangement is slightly different: each subnet is associated with an entire region, but when you provision a new instance, you can select an availability zone in the subnet's region.

The association of regions with subnets in these providers make subnets the means by which you take geography into account when architecting the network topology for your clusters. There is a need to strike a balance between the fast, open communication possible between instances in a single subnet with the availability and reliability benefits of distributing instances across regions, and therefore across subnets.

---

<sup>1</sup> See ??? for a description of gateway instances.

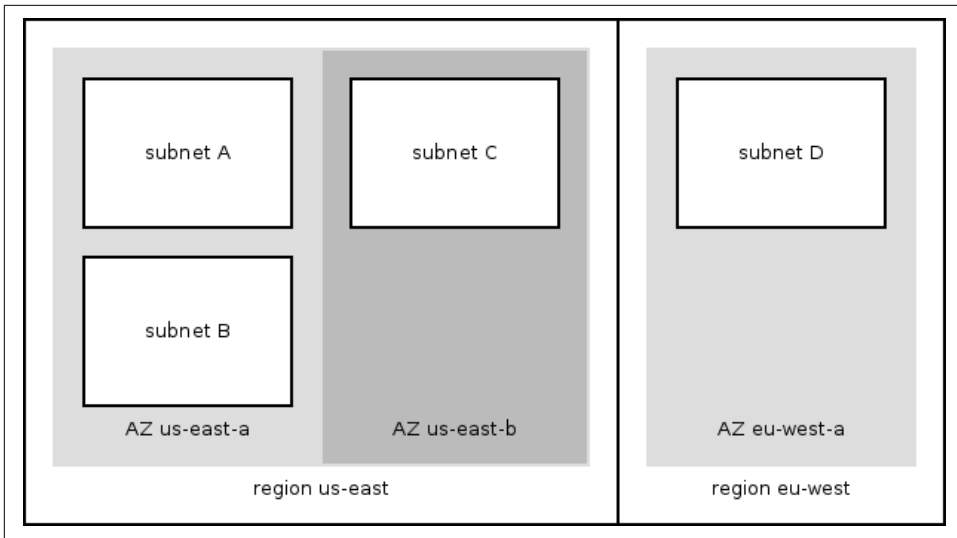


Figure 3-2. A virtual network spanning two regions and multiple availability zones.

Region determination works differently in Azure. With this provider, each subnet and, in turn, each virtual network is associated with a resource group, and a resource group specifies the region for all its resources. So, setting up a cluster that spans regions is somewhat more challenging in Azure, since you will need multiple resource groups, which spreads out management.

??? goes into detail about spanning cluster across availability zones and regions. The general advice is to never span regions, and rarely even span availability zones, due to the impact to performance and the high cost of data transfer, given the large amount of intracluster traffic that Hadoop generates. An architecture that keeps clusters confined to single regions, and even single availability zones, is much more cost-effective.

## Routing

Cloud networking is about much more than placing instances in the right IP address ranges in the right geographic regions. Instances need paths to follow for communication with other instances and with the world outside the cloud provider. These paths are called *routes*.

From the point of view of an instance, there are several possibilities for where a route leads. The shortest and simplest path is back to the instance's own subnet. There are also usually routes that lead to other subnets in the instance's virtual network. Some other routes lead outside the network, either to other virtual networks, or completely outside the cloud provider.

A route is composed of an IP address range and a destination. The route declares that a resource whose IP address falls within its range can be reached by communicating with the route's destination. Sometimes the destination is the desired resource itself, a direct route; but sometimes it is a device, specialized instance, or cloud provider feature that handles establishing communication. In general, that sort of intermediary destination is called a *gateway*.

Here are some examples of routes.

- For IP addresses in the CIDR range 192.168.128.0/18, the destination is subnet C.
- For IP addresses in the CIDR range 10.0.0.0/8, the destination is the corporate VPN gateway.
- For any IP address (CIDR 0.0.0.0/0), the destination is the internet gateway.

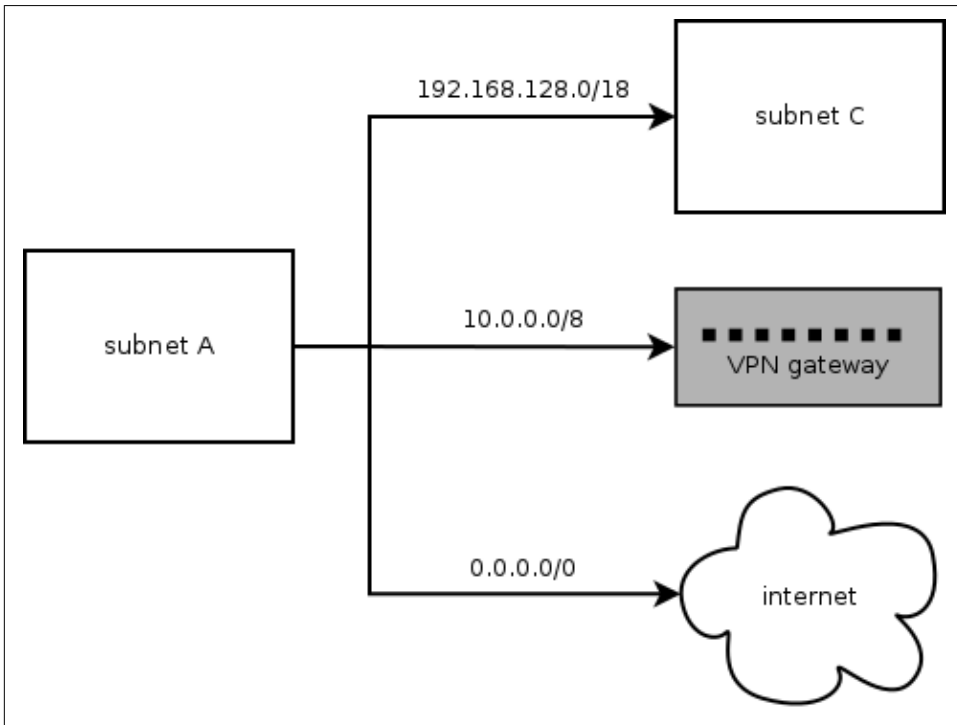


Figure 3-3. Routes leading from a subnet to various destinations.

A cloud instance has a set of routes to look through, which are arranged into a *route table* or *routing table*. Given an IP address to communicate with, the route table is consulted, and the best match for the destination IP address is used. Sometimes there is only one route that satisfies the need, but sometimes there are multiple. In that case, generally, the route that most specifically matches the IP address is chosen.

Suppose that an instance has an associated route table listing the three example routes as shown in [Figure 3-3](#). When a process on the instance attempts to initiate communication with 10.0.0.126, the instance's networking system consults the route table and looks for the best match. The IP address does not fall within the range for subnet C, so that route is discarded. The VPN route and the internet route both match; however, the VPN route is a better match, so that route is chosen.

If there is no match at all, then network communication will fail. That is why it is typical for there to be a catch-all (or default) route for CIDR 0.0.0.0/0 that leads to the internet, the idea being that any IP address not accounted for must be outside the cloud provider.



Designing a cloud network architecture can appear daunting. A cloud provider gives you a lot of power and flexibility, but that carries complexity as well. Fortunately, when you create your first virtual network, the cloud provider sets up a reasonable default configuration for networking so you can get started quickly. For exploratory use the configuration is often acceptable, but before long you will want to look at routing and security rules to ensure they are set up for what you need, such as your first Hadoop cluster.

Routing is an important factor in building out a functioning Hadoop cluster. The daemons that comprise each service, like HDFS and YARN, need to be able to connect to each other, and the HDFS datanodes in particular need to be available for calls from pieces of other services. If all of a cluster's daemons are confined to a single subnet, then the cloud provider's default routing is enough; some providers can even handle routing across subnets automatically or with their defaults. For reaching out farther, such as across VPNs or to the internet, then it usually becomes necessary to define some routes, as the defaults start out restrictive for the sake of security.

Each cloud provider provides routing services in a different way.

## Routing in AWS

In AWS, a route table is an independent object which is associated with VPCs and subnets. Each VPC has a route table which is used as the default for subnets which do not have their own route tables.

The destination for each route is termed a *target*. There are a variety of targets available, some of which are described below.

- A “local” target points to the virtual network of the communicating instance. This covers not only that instance's own subnet, but other subnets in the same VPC.

- An internet gateway target provides access to the internet, outside the cloud provider. When a subnet has a route to the internet, it is called a *public subnet*; without one, it is called a *private subnet*. See ??? for more detailed descriptions.
- A virtual private gateway links to your corporate network's VPN device, allowing access to resources within that network. To establish this connection, you must define a *customer gateway* in AWS representing the VPN device, create a virtual private gateway that links to it, and then define the IP address ranges covered by the gateway.
- A VPC peering connection allows for communication between VPCs using just private IP addresses.
- A NAT gateway provides access to the internet for private subnets. The gateway itself resides in a public subnet.

## Routing in GCP

In GCP, each route is associated with a network, so all instances in the network may have use of it. Routes and instances are associated by tags: if a route has a tag, then it is associated with any instance with a matching tag; if a route has no tag, it applies to all instances in the network.

All of the routes defined for a network form the network's *route collection*, while all of the routes that are associated with an instance form that instance's routing table.

The destination for each route is termed its *next hop*. There are a variety of destinations available, some of which are described below.

- A subnet, or portion of a subnet, can be designated as the next hop by providing a CIDR address for its private IP addresses.
- An internet gateway URL for the next hop provides direct access to the internet, as long as the source instance has an external (public) IP address.
- The URL or IP address of a single instance can be the next hop. The instance needs to be configured with software that can provide connectivity to the ultimate desired destination. For example, the instance could use Squid as a proxy or perform NAT using iptables to provide internet access.

GCP provides a service called Cloud VPN to help manage connectivity between your corporate VPN device and your instances in virtual networks, as well as between virtual networks in GCP itself. A VPN gateway leading to a VPN tunnel is another possible next hop for a route.

## Routing in Azure

In Azure, a route table is an independent object which is associated with subnets. A route table may be associated with multiple subnets, but a subnet can have only one route table.

Azure provides *system routes* for common needs, which are usually comprehensive enough so that you do not need to define a route table at all. For example, system routes direct network traffic automatically within a subnet, between subnets in a virtual network, and to the internet and VPN gateways. If you define a route table for a subnet, its routes take precedence over system routes.

The destination for each route is termed its *next hop*. There are a variety of destinations available, some of which are described below.

- The local virtual network can be specified as the next hop for traffic between subnets.
- A virtual network gateway or VPN gateway for the next hop allows traffic to flow to other virtual networks or to a VPN.
- Naming the internet as the next hop provides direct access to the internet.
- A null route or black hole route can be used as the next hop to drop outgoing traffic completely.

## Network Security Rules

If routing builds the roads for traffic between your instances, then security rules define the laws the traffic must obey. Cloud providers separate the definitions of the connections between instances from the definitions of how data may flow along those connections.

In a way, routing is a coarse-grained security measure: if there is no route defined between an instance and some destination, then absolutely no traffic can pass between them. When a route is established, however, then security rules provide a way to allow some kinds of traffic and disallow others.

As with routing, each cloud provider provides network security in different ways, but they share common concepts.

## Inbound vs. Outbound

*Inbound* rules control traffic coming to an instance, while *outbound* rules control traffic leaving an instance. Most of the time, you will find yourself focusing on inbound rules, and allowing unrestricted traffic outbound. The implication is that you trust



the activity of the instances that you yourself control, but need to protect them from traffic coming in from outside, particularly the internet.

## Allow vs. Deny

An *allow* rule explicitly permits some form of network traffic, while a *deny* rule explicitly blocks it. If an allow rule and a deny rule conflict with each other, then usually the deny rule wins out. A common pattern is to establish an allow rule with a broad scope, and then use deny rules to pick out exceptions; for example, you could allow HTTP access from everywhere with one rule, and add deny rules that block IP ranges that are known to exhibit bad behaviors.

Some security rule structures do not use allow rules at all. Instead, they start from an initial implicit state of denying all traffic, and you add allow rules for only what you want to permit.

## Network Security Rules in AWS

AWS provides two main mechanisms for securing your VPC.

### Security Groups

The most common mechanism used is *security groups*. A security group provides a set of rules that govern traffic to and from an instance. Each instance can belong to one or several security groups; a VPC also has a default security group which applies to instances that aren't associated with any groups themselves.

Each rule in a security group only allows traffic. If none of the security groups for an instance allow a particular kind of traffic, then that traffic is denied by default.

A rule in a security group can apply to either inbound traffic or outbound traffic. An inbound rule allows traffic into an instance over a protocol (like TCP or UDP) and port or port range from either another security group or a range of IP addresses. Similarly, an outbound rule allows traffic out from an instance to either another security group or to a range of IP addresses. Here are some examples of typical inbound and outbound security group rules.

- If you are running a web server on an instance, an inbound rule for TCP port 80 can allow access from your IP address, or the IP range for your corporate network, or the entire internet (0.0.0.0/0).
- To allow SSH access to an instance, an inbound rule should permit access for TCP port 22. It's best to restrict this rule to your own IP address, or those in your network.
- If a process running on an instance will need to access a MySQL server elsewhere, an outbound rule over TCP port 3306 will allow it. The destination could

be the IP address of the MySQL server or, if the server is running in EC2, the server's security group.

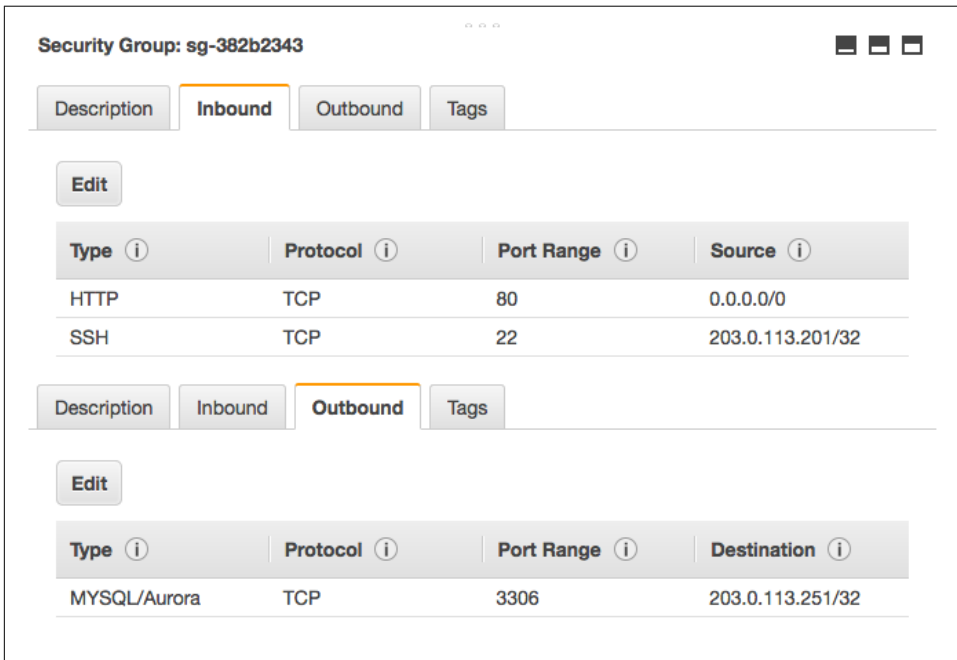


Figure 3-4. A security group with some example rules. (This figure is a composite view showing both the Inbound and Outbound tabs.)

Each VPC comes with a simple, default security group that allows outbound access to anywhere, but inbound access only from other instances in the same security group. This means that you need to set up SSH access from your local network before you can access instances provisioned there.

One convenient feature of security groups is that you only need to allow one side of a two-way connection. For example, it is enough to allow TCP port 80 inbound for a web server; since requests are allowed to flow in, AWS automatically permits responses to flow back outbound from the web server. This feature is not true of the other main mechanism for securing VPCs, network ACLs.

## Network ACLs

*Network ACLs* are a secondary means of securing a VPC. Like security groups, they are comprised of a set of rules that govern network traffic. Unlike security groups, a network ACL is associated with a subnet, not individual instances. A subnet may only have one network ACL, or else it falls back to its VPC's default network ACL.

A network ACL rule can either allow or deny traffic. While all of the rules in a security group apply to every network access decision, the rules in a network ACL are evaluated in a numbered order, top to bottom, and the first matching rule is enforced. If none of the rules match, then the fixed, final default rule in every network ACL denies the traffic.

Each network ACL rule is an inbound rule or outbound rule, as with security group rules. A rule applies to a protocol and port range, but sources and destinations are only specified as IP address ranges, not as security groups.

**Figure 3-5** lays out a simple network ACL that allows limited inbound SSH access and HTTP access, but no other network traffic.

**acl-337b3b54 | example**

Summary

**Inbound Rules**

Outbound Rules

Subnet Associations

Tags

Allows inbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.

Edit

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH (22)	TCP (6)	22	192.168.0.0/16	ALLOW
200	HTTP (80)	TCP (6)	80	192.168.0.0/16	ALLOW
220	HTTP (80)	TCP (6)	80	10.0.0.0/16	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Summary

Inbound Rules

**Outbound Rules**

Subnet Associations

Tags

Allows outbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.

Edit

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
100	Custom TCP Rule	TCP (6)	1024-65535	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

*Figure 3-5. An ACL with some example rules. (This figure is a composite view showing both the Inbound and Outbound tabs.)*

The inbound rules allow only SSH access from one IP address range and HTTP port 80 access from two IP address ranges. Any other inbound network access is blocked by the default final deny rule.

The outbound rules allow any traffic over non-privileged TCP ports. This is necessary to permit outbound traffic for SSH and HTTP connections. Unlike security groups, network ACLs require you to allow both sides of two-way connections. Since it can be unpredictable what port a requesting process may use to connect out from its host, the network ACL rule here permits a wide range of ports.

To illustrate, here is an example of an HTTP client outside the virtual network performing an HTTP request to a server running inside the network. The simple ACL defined above gates both the incoming request and outgoing response.

*Inbound request from 10.1.2.3:12345 to port 80*

- rule 100: does not apply (port range)
- rule 200: does not apply (source CIDR)
- rule 220: applies, so access is allowed

*Outbound response from port 80 to 10.1.2.3:12345*

- rule 100: applies, so access is allowed

Each VPC comes with a default network ACL which allows all traffic inbound and outbound. So, by default, your VPC does not make use of a network ACL for security, but it is still available for a second line of defense.

## Network Security Rules in GCP

GCP supports *firewall rules* for governing traffic to instances in a network. Firewall rules are associated with a network itself, but they can apply to some or all of the instances in that network.

Each firewall rule only allows traffic. If none of the firewall rules for a network allow a particular kind of traffic, then that traffic is denied by default.

A firewall rule controls inbound traffic only. You can control outbound traffic from an instance using network utilities installed on the instance itself.

You can specify the source a firewall rule applies to as either a range of IP addresses, a subnet in the network, or an instance tag. When a subnet is specified, then the rule applies to all of the instances in that subnet as sources. An instance tag limits the applicability of a firewall rule to just instances with that tag.

Each firewall rule names a protocol (like TCP or UDP) and port or port range on the destination instances that it governs. Those instances can be either all of the instances in the network, or just instances with another instance tag, called a target tag.

Here are some examples of typical firewall rules. There are shown with some others in [Figure 3-6](#).

- If you are running a web server on an instance, a rule for TCP port 80 can allow access from your IP address, or the IP range for your corporate network, or the entire internet (0.0.0.0/0). To narrow down where the firewall rule applies, you can tag the web server instance as, say, “webserver”, and provide that tag as the target tag for the rule.
- To allow SSH access to an instance, a rule should permit access for TCP port 22. It’s best to restrict this rule to your own IP address, or those in your network.



<input type="checkbox"/> Name ^	Source tag / IP range / Subnetworks	Allowed protocols / ports	Target tags	Network
<input type="checkbox"/> default-allow-icmp		icmp	Apply to all targets	default
<input type="checkbox"/> default-allow-internal	10.128.0.0/9	tcp:0-65535, 2 more ▾	Apply to all targets	default
<input type="checkbox"/> default-allow-ssh		tcp:22	Apply to all targets	default
<input type="checkbox"/> example-allow-http	0.0.0.0/0	tcp:80	webserver	default
<input type="checkbox"/> example-allow-ssh	203.0.113.201/32	tcp:22	Apply to all targets	default

Figure 3-6. Some firewall rules. (Some source IP ranges are obscured.)

The default network that GCP supplies for you comes with a small set of firewall rules that allow all traffic within the network as well as SSH (TCP port 22), RDP (port 3389) and ICMP from anywhere. This is a reasonable default behavior, although it makes sense to adjust the rules to limit sources to just your own IP address or your own network. Any new networks you create, however, do not start out with any firewall rules, and so absolutely no inbound access is permitted. It is up to you to build out the necessary firewall rules to gain access.

One convenient feature of firewall rules is that you only need to allow one side of a two-way connection. For example, it is enough to allow TCP port 80 inbound for a web server; since requests are allowed to flow in, GCP automatically permits responses to flow back outbound from the web server.

There are a few automatic firewall rules that are enforced on all networks. Here are some of them.

- TCP port 25 (SMTP) is always blocked outbound from your network.
- TCP ports 465 and 587 (SMTP over SSL) are also always blocked outbound, except to SMTP relay services hosted on Google Apps.
- Network traffic using a protocol besides TCP, UDP, or ICMP is blocked unless the Protocol Forwarding feature of GCP is used to allow it.



Check the latest [GCP documentation](#) for ways to send email from its instances, such as SMTP relays, that involve third-party email providers.

One final security rule deserves mention here. If an instance does not have a external IP address assigned to it, then it is not granted access to the internet. This rule is enforced even if a network route provides a path to an internet gateway URL. To reach the internet from such an instance, it's necessary to go through a gateway, using either NAT or a VPN.

## Network Security Rules in Azure


Azure provides *network security groups* for controlling traffic into and out of either subnets or individual virtual machines through their network interfaces. A virtual machine can be subject to its subnet's network security group as well as its own.

A network security group holds a set of rules, each of which controls either inbound traffic or outbound traffic. An inbound rule allows or denies traffic into an instance over a protocol (like TCP or UDP) and port or port range from either a range of IP addresses, a default tag (defined below), or all sources. Similarly, an outbound rule allows or denies traffic out from an instance to either a range of IP addresses, a default tag, or all destinations.

A *default tag* is a symbolic representation for a set of IP addresses. For example, the virtual network tag stands in for the local virtual network and those connected to it. The internet tag represents the internet, outside of Azure's infrastructure and connected VPNs.

Here are some examples of typical inbound and outbound security group rules.

- If you are running a web server on an instance, an inbound rule for TCP port 80 can allow access from your IP address, or the IP range for your corporate network, or the entire internet using the internet default tag.
- To allow SSH access to an instance, an inbound rule should permit access for TCP port 22. It's best to restrict this rule to your own IP address, or those in your network.
- If a process running on an instance will need to access a MySQL server elsewhere, an outbound rule over TCP port 3306 will allow it. The destination could be the IP address of the MySQL server.

2 Inbound security rules 					
PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
100	allow_http	Internet	Any	HTTP (TCP/80)	Allow
200	allow_ssh	203.0.11...	Any	SSH (TCP/22)	Allow


1 Outbound security rule 					
PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
100	allow_mysql_out	Any	203.0.113.251...	MySQL (TCP/3306)	Allow

Figure 3-7. A network security group with some example rules.

Rules are evaluated in priority order to determine which one holds sway. A lower number priority on a rule indicates a higher priority.

Every network security group has a default set of rules, which are lower priorities than any user-defined rules. They allow, among other things, all traffic from the same virtual network and all outbound traffic to the internet, but deny inbound traffic from anywhere but the virtual network. The rules can be overridden with user-defined rules.

## Putting Networking and Security Together

As you have seen, there is a lot to think about when it comes to networking and security in a cloud provider. Getting started with them can feel like jumping into the deep end of a pool, or being dropped into a foreign land without a map. Here are some pointers to getting rolling.

Cloud providers do try to start you out with a sensible initial arrangement: a single virtual network with one or a few subnets, and default routing and security rules applied. Of all the concerns, routing tends to require the least amount of attention, as defaults and fallbacks define almost all of the necessary connections.

For small to medium Hadoop deployments, a single virtual network usually suffices. As described in the beginning of this chapter, it is useful to think of each virtual network as a container for your clusters. With subnets to provide any necessary divisions

or regional variations, and ample IP addresses available, you may find you can go a long time before needing to define an entirely new network.

Routing and security rules become more important once traffic needs to be sent to or received from outside a virtual network. Keeping Hadoop clusters confined to single subnets or, at worst, single virtual networks eliminates most of the need to define routes and security rules. One important exception is allowing SSH access to some number of instances, which is described in following chapters about getting started with each cloud provider. Another is opening up ports for applications running alongside clusters, or for web interfaces of Hadoop components. For these exceptions, the process is typically only defining a route if necessary and declaring a security rule that allows access.

more?

## What about the Data?

The purpose here of creating cloud instances, networking them together, and establishing routes and security rules is to stand up Hadoop clusters, and the purpose of these clusters is to work on data. The data moves through the network between instances, but where is it stored?

As you would expect, cloud providers offer ranges of storage options that include disks, databases, general object storage, and other services. Understanding how these storage options can be used is just as important for creating effective clusters as understanding networking and security.



# Setting up in AWS

In this chapter you'll create a simple Hadoop cluster running in EC2 (Elastic Compute Cloud), the service in Amazon Web Services (AWS) which enables you to provision instances. The cluster will consist of basic installations of HDFS and YARN, which form the foundation for running MapReduce and other analytic workloads.

This chapter assumes you are using a Unix-like operating system on your local computer, such as Linux or Mac OS X. If you are using Windows, some of the steps will vary, particularly those about working with SSH.

## Prerequisites

Before you start, you will need to have an account already established with AWS. You can register for an account for free.

Once you are registered, you will be able to log in to the *AWS console*, a web interface for using all of the different services under the AWS umbrella. When you log in, you are presented with a dashboard for all of those services. It can be overwhelming, but fortunately, for this chapter you only need to use one service: EC2.

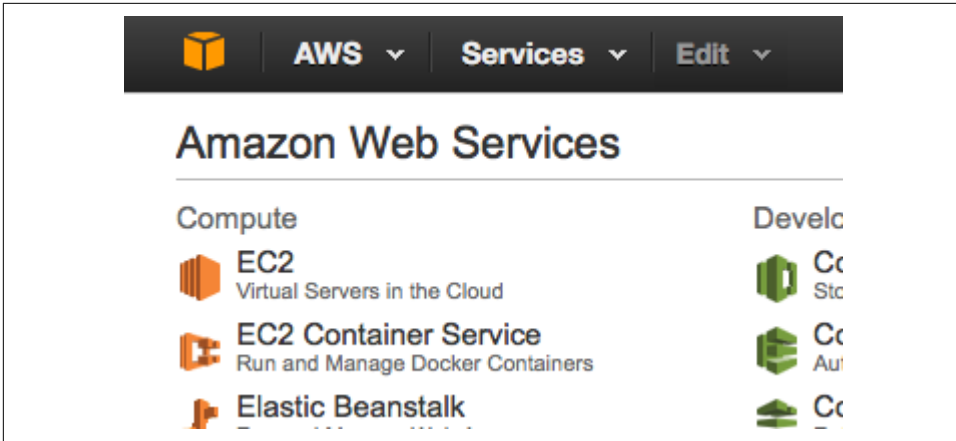


Figure 4-1. EC2 on the AWS dashboard



Over time, Amazon will update the arrangement of the console, and so the exact instructions here may become inaccurate.

Notice at the top of the page a dropdown menu with a geographic area selected, such as “N. Virginia”, “N. California”, or “Ireland”. This indicates the AWS region where you are currently working. If the selected region is not the one you prefer, go ahead and select a different one using the dropdown. It is generally a good idea to use a region that is close to your location.

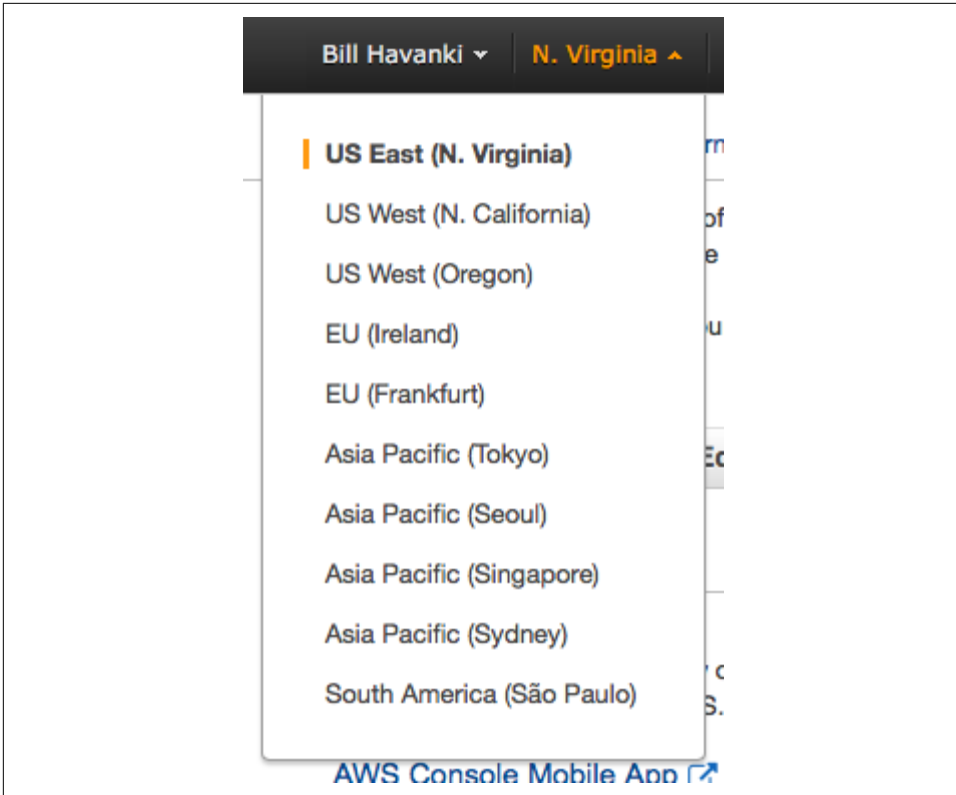


Figure 4-2. A dropdown menu for the AWS regions

## Allocating Instances

In this section, you will launch the instances that will comprise a Hadoop cluster. You will also ensure that they have the right networking connections, and that you can connect to them over SSH.

### Generating a Key Pair

Before provisioning instances in EC2, you must create a *key pair*. A key pair consists of a public key and a private key, to be used for SSH communication with your instances. If you are already familiar with asymmetric encryption technologies such as SSL/TLS, you will be comfortable with EC2 key pairs.

You can have AWS generate the key pair for you and download the private key. As an alternative, you can use a client-side tool like OpenSSL to create the key pair yourself, and then upload the public key to AWS. For now, you'll let AWS do the work.

To have AWS generate a key pair, log in to the AWS console and select EC2. A menu will appear on the left side of the page with options for working in EC2. Select “Key Pairs” in the menu. The main area of the page will show that you have no key pairs for the region yet. Press the “Create Key Pair” button to start the process of creating a new key pair, and follow the instructions.

When you are done, you will receive a PEM file that contains a private key. The contents of the file start with “BEGIN RSA PRIVATE KEY”. This is the key file you will use to SSH to your provisioned instances. Save the file in a safe place, using file permissions on your local computer to protect it (for example, make it readable only by your account). A great place to store it is in the hidden `.ssh` directory in your home directory, which is normally used by OpenSSH for key storage and other sensitive files.

```
$ mv keyfile.pem ~/.ssh
$ chmod 600 ~/.ssh/keyfile.pem
```

## Launching Instances

You’re now ready to launch instances. For this simple cluster, you’ll launch four instances: one “manager” and three “workers”. The manager instance will host the HDFS namenode and the YARN resource manager, while the workers will host the HDFS datanodes and YARN node managers. A minimum of three workers is recommended, since the default HDFS replication factor is three.

### The Manager Instance

Let’s start by launching the manager instance. Select “Instances” from the EC2 menu. (Pick EC2 from the set of AWS services if you need to.) You’ll see that you don’t have any running instances yet; you’ll change that. Press the “Launch Instance” button to start the process.

In the first step, you need to choose an image to base your instance on. In AWS, an image is called an *Amazon Machine Image*, or *AMI*. A “Quick Start” set of AMIs is presented to you. The AMIs cover popular operating systems and architectures; they are updated periodically as new recommended images are made available. Find an AMI for Ubuntu Linux that uses virtualization type HVM and select it.

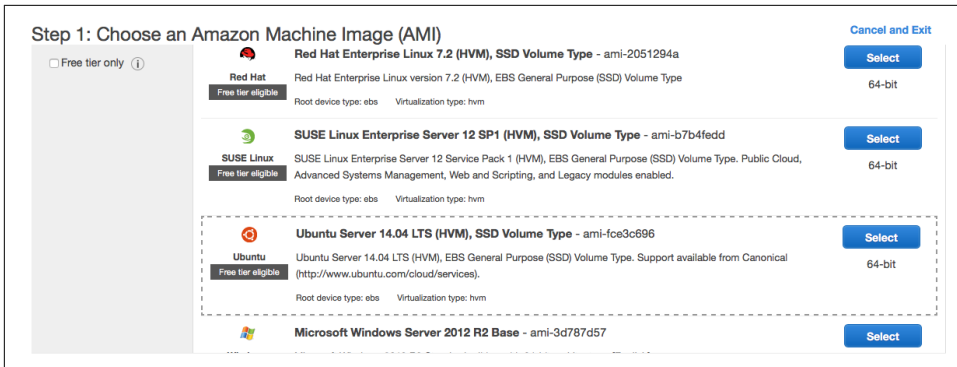


Figure 4-3. Some AMI choices

## Free Tier

For the first year that you have an AWS account, you are eligible for the *free tier*, which lets you provision small instances on free operating systems without any cost. The free tier is great for exploring how to use AWS, but the instance types that fall within it are, sadly, too underpowered to host a Hadoop cluster that can do more than trivial work.



You do not need to use Ubuntu to deploy Hadoop in the cloud. It was chosen for the instructions here because it is popular and widely supported, across all cloud providers. The steps for using a different Linux distribution can differ, especially in the system tools available and in package management.

Next, select an instance type for the manager. A manager instance often needs more powerful specifications than a worker. For this cluster, select an instance type in the General Purpose family with at least four vCPUs and at least 8 GB of memory<sup>1</sup>. Over time, EC2 introduces new instance types with faster processing, better or more memory, or better storage capabilities, and these new types become preferred over older ones. Unless you have particular requirements that only an older instance type satisfies, you should favor new instances types.

<sup>1</sup> If you just created your AWS account, you may discover that you are not permitted to use such a powerful instance type right away. If so, go ahead and use a free tier instance type that at least has the minimum memory needed. The cluster will still work for experimentation.

Step 2: Choose an Instance Type							
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	m4.large	2	8	EBS only	Yes	Moderate
<input checked="" type="checkbox"/>	General purpose	m4.xlarge	4	16	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.4xlarge	16	64	EBS only	Yes	High

Figure 4-4. Selecting an EC2 instance type

Instead of pressing the tempting “Review and Launch” button, you need to work through the full set of steps for launching an instance to fully configure it. Press the “Next: Configure Instance Details” button to review networking choices and other information.

### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances 1 [Launch into Auto Scaling Group](#)

Purchasing option ☐ Request Spot instances

Network vpc-e23a3c86 (172.31.0.0/16) (default) [Create new VPC](#)

Subnet No preference (default subnet in any Availability Zone) [Create new subnet](#)

Auto-assign Public IP Use subnet setting (Enable)

Placement group No placement group

IAM role None [Create new IAM role](#)

Shutdown behavior Stop

Enable termination protection ☐ Protect against accidental termination

Monitoring ☐ Enable CloudWatch detailed monitoring  
[Additional charges apply.](#)

EBS-optimized instance ☒ Launch as EBS-optimized instance

Tenancy Shared - Run a shared hardware instance  
[Additional charges will apply for dedicated tenancy.](#)

Figure 4-5. EC2 instance details

There is a lot to digest here, but here are some highlights.

- The number of instances to launch defaults to one. That is fine for now, because only one manager is necessary for a simple cluster.
- A VPC has already been selected. Your AWS account is automatically set up with a VPC upon creation; if you are using a long-standing, shared account, there may be a few VPCs to choose from.
- The shutdown behavior defines what happens if you issue a system shutdown within your instance (for example, `/sbin/shutdown`). The default of “Stop” leaves the instance available in your account to be started again, but selecting “Termi-

nate” causes the instance to instead be destroyed. You should leave “Stop” selected.

- You can enable termination protection on instances so that they cannot be terminated without explicitly disabling the protection via the console as an extra step.

Since the cluster will have multiple instances talking to one another, configuration will be easier and performance will be much better if they are all close to each other in the network. So, select a specific subnet from the list of available subnets. A *subnet* is a segment of the overall address space in a VPC that resides within a single availability zone. Remember the subnet that you choose now, so that the worker instances can be provisioned there as well.

Press “Next: Add Storage” to see options for adding storage to the new instance.

**Step 4: Add Storage**

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted	
	Root	/dev/sda1	snap-f70def0	8	General Purpose SSD (GP2)	24 / 3000	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Figure 4-6. EC2 instance storage configuration

By default, an EC2 instance comes with a modest amount of dedicated storage mounted as its root volume. Increase the amount, if necessary, to at least 30 GiB, so that there is ample room for installing what you need for your cluster. Resist the temptation to add more storage than you need: you will be charged per GB provisioned, not per GB utilized. Then press “Next: Tag Instance” to see the tag entry form.

**Step 5: Tag Instance**

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)
Name	

[Create Tag](#) (Up to 10 tags maximum)

Figure 4-7. Setting EC2 instance tags

Tags are simple name-value pairs that are attached to instances. They are useful for organizing instances and noting their name, purpose, owner, or any other basic information. For this instance, enter a tag with key “Name” and value “manager”. Press “Next: Configure Security Group” to set up security for the new instance.

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group  
☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source
SSH	TCP	22	Anywhere 0.0.0.0/0

Figure 4-8. The default security group selection for EC2 instances

Here, you will define the permissions for network communications to and from the cluster through a security group. Start by selecting the “Create a new security group” radio button, and then entering “basic-security” for the group name. Supply any description you like.

The new security group defaults to having one incoming rule, allowing SSH access from any IP address, represented by the CIDR<sup>2</sup> 0.0.0.0/0. While this can be acceptable for a case such as this where you are only trying things out, you should instead restrict access to only an appropriate range of IP addresses.

To restrict SSH access to your IP address alone, either replace the default CIDR address by selecting “Custom IP” and enter your IP address, or just select “My IP” to have your address autodetected. To instead restrict access to a range of IP addresses (for example, your address may change over time, or you may switch local computers), select “Custom IP” and enter the appropriate CIDR address.



If you are on a local network that accesses the internet through a router performing network address translation (NAT), then be sure to use the IP address assigned to the router. The “My IP” autodetection should select the correct IP address.

<sup>2</sup> [https://en.wikipedia.org/wiki/Classless\\_Inter-Domain\\_Routing](https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing)



### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group  
☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source
SSH	TCP	22	Anywhere 0.0.0.0/0

Figure 4-9. Updated security group configuration for EC2 instances

Press the “Review and Launch” button to see an overview of the instance you are about to launch. The chosen AMI, instance type, security group, and other details are available to look over. When you are satisfied, press the “Launch” button to complete the last step.

Before EC2 proceeds to launch your instance, you will be asked for a key pair to use for SSH access. Since you already created a key pair, select its name and press “Launch Instances”.

Your new instance will start to launch. You can monitor its progress by returning to the AWS console and selecting “Instances” from the EC2 menu. Once the instance state is shown as “running”, you can attempt to connect to it over SSH, using the private key from your chosen key pair to authenticate. The username to use depends on the AMI you have chosen; sometimes “root” will work, but often it will be a different username; for Ubuntu, you can try “ubuntu” as the username. You can use either the public DNS name or public IP address to reach the instance; these can be found in the instance details when you select the manager instance from the list of running instances.



The AWS console can help you with your SSH command line for connecting to an instance. Select Instances from the EC2 menu, select the row for the instance you want to connect to, and press the “Connect” button. A dialog box will suggest an SSH command line, including a username that may work.

## The Worker Instances

Once you can connect to the manager instance, you’ve completed the first step towards standing up a cluster. Now, repeat the above steps to launch the three worker instances. The procedure is almost the same, but there are important changes.

- You can choose a less powerful instance type for workers: for example, an instance type with only 2 vCPUs.
- When configuring instance details, change the “Number of Instances” to 3.

- Be sure to select the same subnet as the one hosting the manager instance. As mentioned above, this will make configuration of the cluster much easier.
- Use the value “worker” for the Name tag.
- Instead of creating a new security group, select the same security group that you created for the manager instance.

After the worker instances are launched, make sure you can SSH to each of them as well. At this point you have a set of instances that is able to run a Hadoop cluster.

## Securing the Instances

Some of the work to secure the new instances was done when the security group for them was defined, but there is a little more to do.

If you took the recommended step of locking down SSH access in the “basic-security” security group created while allocating instances, then SSH between instances will not work. This is because the security group only allows SSH from your IP address (or IP address range), and nowhere else. So, the security group must be updated to allow wider SSH access. In fact, there will be more than just SSH traffic between instances once the cluster is running, so the best way to allow it all is to open up inbound traffic from anywhere in the security group itself.

Start by selecting “Security Groups” from the EC2 menu. Select the row for the “basic-security” group, and then select the “Inbound” tab for the group. Press the “Edit” button, and then “Add Rule” in the dialog box that appears. For the Type of the new rule, select “All traffic”, and enter the security group ID or name in the “Source” field. Press “Save”, and the security group will be updated. At this point, it should be possible to SSH between any two instances in the security group.

## Next Steps

At this point, there are instances destined to host a cluster running in EC2. To pause here, you can stop the new instances by selecting them in the list of running instances and then using the “Actions” button to select “Instance State”, and then “Stop”. You can start them later by using “Start” in the same menu.

Otherwise, proceed to [Chapter 5](#) to install Hadoop and configure it, and then try it out with some basic MapReduce jobs.

---

# Standing up a Cluster

Now that you have instances up and running in the cloud provider of your choice, they can be set up to run a Hadoop cluster. If you don't have instances at the ready and want to follow along, then go back to [Chapter 4](#) for AWS, [???](#) for GCP, or [???](#) for Azure first, and then return here.

## The JDK

Hadoop requires a Java runtime to work, and so Java must be installed on each of your new instances. A good strategy is to use the operating system package management capability already on the instances to do so, e.g., `yum` on Red Hat Linux, `apt` on Ubuntu. Cloud providers ensure that these capabilities work within their infrastructures, sometimes even providing local mirrors or gateways to help.

Here are suggested packages to install for some operating systems. As new versions of Java are released, the package names will change.

*Table 5-1. Suggested Java Packages*

Debian or Ubuntu	<code>openjdk-8-jdk</code> or <code>openjdk-7-jdk</code>
Red Hat or CentOS	<code>java-1.8.0-openjdk</code> or <code>java-1.7.0-openjdk</code>

Instead of using a package available natively for your operating system, you can install an Oracle JDK by downloading an installation package directly from Oracle. Since you have root access to your instances, you are free to use whatever means you prefer to install Java.

After you have installed Java, make note of where the Java home directory is (i.e., what the `JAVA_HOME` environment variable should be set to). You will need to know this location later.

# Hadoop Accounts

While Hadoop can run under the root account, it is better security practice to use non-privileged accounts. For this simple cluster, create two ordinary user accounts on each instance for running HDFS and YARN. These instructions will assume that the usernames for the account are “hdfs” and “yarn”. They will both belong to a new “hadoop” group. For even better security, instead of creating passwords for the new accounts, use an SSH key pair. One option is to copy in the `authorized_keys` file from the standard login account for the instance; that way, you can use the same private key to connect via SSH from your local computer to any of the accounts on the instances.

```
$ sudo groupadd hadoop
$ for u in hdfs yarn; do
> sudo useradd -G hadoop -m -s /bin/bash ${u}
> sudo mkdir /home/${u}/.ssh
> sudo cp ~/.ssh/authorized_keys /home/${u}/.ssh
> sudo chmod 700 /home/${u}/.ssh
> sudo chown -R ${u} /home/${u}/.ssh
> done
```

## Passwordless SSH

The Hadoop distribution includes some helpful utility scripts that can connect to each instance in your cluster to start and stop everything for a service at once. To use these scripts, passwordless SSH must be established from the “hdfs” and “yarn” accounts on the manager instance to the same accounts on each of the worker instances, as well as hopping from the manager instance back to itself.

While your cloud provider key pair can be used for passwordless SSH, it’s better to keep its private key from being copied too widely. So, instead, generate new default SSH keys on the manager instance under each account, and then transfer the public keys to the corresponding accounts on the worker instances.

```
# on manager:
$ ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -N ''
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

# on each worker:
$ cat >> ~/.ssh/authorized_keys
# copy and paste public key contents
^D
```

The commands above accomplish transferring the public keys by just copying and pasting their contents through your SSH sessions. Another option, which is automatable, is to copy the public keys using SCP back to your local computer, and then copy

them again from there to each worker instance. On a larger system, management tools like Chef or Puppet could be used to automatically distribute keys.

Now that the public keys have been distributed, connect via SSH from the “hdfs” and “yarn” accounts on the manager to the same accounts on each instance, including the manager itself, using each instance’s private IP address. This is not only a useful check, but also gives you the chance to accept the key fingerprint for each worker and have it recorded in the SSH known hosts files of the manager accounts. Without doing so, automated scripts that use SSH might get stuck waiting for interactive confirmation of a fingerprint.

If the connections fail, verify the security rules governing the instances, and make sure to use the private keys generated for each Hadoop account, not the private key set up with the cloud provider.

## Hadoop Installation

For now, the Hadoop cluster will only contain the basic Hadoop services HDFS and YARN. These instructions are based on the [standard cluster setup instructions](#) and use the standard binary distribution from Apache. If you already have a tried-and-true set of steps for Hadoop installation, or if you prefer to use a customized or bundled distribution from a Hadoop vendor, you may be able to adapt these instructions.

To start, download a binary Hadoop distribution from [hadoop.apache.org](http://hadoop.apache.org) to each of your instances, under the standard login account. You can download to your local computer and then use SCP to transfer the distribution to each of your instances, or instead use `curl` or `wget` on each instance to download the distribution directly.

Apache uses a mirror system to offer distributions, so the simplest way to find a download URL is to visit [the Hadoop download page](#) to have it select a mirror, and then use the chosen mirror URL for your download command line.

Since multiple user accounts will be running Hadoop components, install it in a commonly accessible location. For this cluster you’ll use `/opt/hadoop` as that common location.

```
$ curl -O http://mirrorhost/path/to/hadoop-x.y.z.tar.gz
$ sudo tar xzfC hadoop-x.y.z.tar.gz /opt
$ sudo ln -s /opt/hadoop-x.y.z /opt/hadoop
```



Example commands in this chapter will use the version number `+x.y.z+` to stand in for the real Hadoop version number. Unless you have a specific need otherwise, you should simply use the latest release.



About now you may notice how often you need to repeat steps for every instance in your cluster. Once you are creating lots of clusters in the cloud, it's a good idea to use a tool that issues commands over SSH to multiple instances at a time. [Chapter 17](#) discusses some options.

Now that Hadoop is installed, its *bin* directory can be added to the PATH environment variable for each of the Hadoop accounts.

```
# as hdfs and as yarn
$ echo "export PATH=\"/opt/hadoop/bin:${PATH}\"" >> ~/.profile
```

## HDFS and YARN Configuration

Once Hadoop is installed across your cluster, it's time to configure it. The procedure here is much like configuring a Hadoop cluster on “real” hardware. Refer to the Hadoop documentation for all of the details, or consult texts like *Hadoop: The Definitive Guide* by Tom White. The instructions here are simple ones, just to get the cluster going. Again, if you are used to configuring Hadoop clusters, go ahead and adapt what you normally do.

Many of these configuration steps require you to use `sudo`. The standard login account for your instances should be able to execute commands using `sudo`.



To avoid needing to type `sudo` before every command, use **`sudo su -`** in the standard instance login account to open a shell as root. Be careful, though, since you will have unrestricted access to the machine.

Unless otherwise stated, the configuration steps should be performed on every cluster node. Some which are only required on the manager node are called out.

## The Environment

Create a script `/etc/profile.d/hadoop.sh` that sets the `HADOOP_PREFIX` environment variable to point to the Hadoop installation. To have the script take effect, either log out and back in again, or source it in your current shell.

```
# as root
% echo "export HADOOP_PREFIX=/opt/hadoop" > /etc/profile.d/hadoop.sh
```



Redirection of a command run under `sudo` happens under the initial account, and not as root. That is why the command above requires you to be logged in as root. An alternative is to use the `tee` utility to write the file, but then discard standard output.

```
$ echo "export HADOOP_PREFIX=/opt/hadoop" | sudo tee /etc/profile.d/hadoop.sh > /dev/null
```

Create a symlink at `/etc/hadoop` that points to `/opt/hadoop/etc/hadoop`, since you will be using that directory for the cluster's configuration.

```
$ sudo ln -s /opt/hadoop/etc/hadoop /etc/hadoop
```

Create the directories `/var/log/hadoop` and `/var/run/hadoop` for Hadoop logs and process ID files, and make them writable only by the Hadoop accounts.

```
$ for d in /var/log/hadoop /var/run/hadoop; do
> sudo mkdir $d
> sudo chgrp hadoop $d
> sudo chmod g+w $d
> done
```

On many Linux distributions, the contents of `/var/run` are stored on a temporary file system and destroyed on reboot, causing the new `/var/run/hadoop` directory to vanish. There are two options to cope with this. First, you can simply use a different directory for process ID files, such as `/opt/hadoop/pids`, which resides on a non-temporary file system; however, this is a non-standard approach. The other option is to employ a system initialization script, in accordance with the Linux distribution you are using, that creates `/var/run/hadoop` on each boot. These instructions assume the latter approach.

Here is an example initialization script for Ubuntu and similar distributions, which can be saved as `/etc/init.d/hadoop`.

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Create Hadoop directories at boot
### END INIT INFO

case "$1" in
start)
    mkdir -p /var/run/hadoop
    chown root:hadoop /var/run/hadoop
    chmod 0775 /var/run/hadoop
    ;;
esac
```

The script can be activated through either of the following commands.

```
# activate with fixed start priority of 98
$ update-rc.d hadoop defaults 98
# activate using dependencies in INIT INFO comments
$ /usr/lib/lsb/install-initd hadoop
```

Next, edit `/etc/hadoop/hadoop-env.sh` and make the following changes.

- Set the path for `JAVA_HOME`. Where this depends on how you chose to install Java. If you used your platform's package manager, try seeing where `/usr/bin/java` or `/etc/alternatives/java` links to.
- Set `HADOOP_LOG_DIR` and `HADOOP_PID_DIR` to point to `/var/log/hadoop` and `/var/run/hadoop`, respectively.

```
export JAVA_HOME=/path/to/jdk
export HADOOP_LOG_DIR=/var/log/hadoop
export HADOOP_PID_DIR=/var/run/hadoop
```

Edit `/etc/hadoop/yarn-env.sh` and set the paths for `YARN_CONF_DIR` and `YARN_LOG_DIR` to point to `/etc/hadoop` and `$HADOOP_LOG_DIR`, respectively. Also append a Java option to the `YARN_OPTS` variable so that YARN daemons prefer to listen over IPv4.

```
export YARN_CONF_DIR=/etc/hadoop
export YARN_LOG_DIR="$HADOOP_LOG_DIR"
YARN_OPTS="$YARN_OPTS -Djava.net.preferIPv4Stack=true"
```

## XML Configuration Files

Edit `/etc/hadoop/core-site.xml` and configure the `fs.defaultFS` property with the address of your manager instance. Use the private IP address of the manager instance in the URL. You can find the private IP address for an instance by locating it in your cloud provider's console.

You can also configure the `hadoop.tmp.dir` property with a different location for Hadoop's temporary file storage. HDFS, for one, uses this location as the basis for where namenode and datanode data is kept. The default value for the directory resides under `/tmp`, which is cleared out when instances are restarted<sup>1</sup>. If you want to be able to stop and start your cluster, you should change this location. The value `/home/${user.name}/tmp` directs Hadoop to use a temporary directory under each Hadoop account's home directory. You could select other persistent locations as well, perhaps those that map to large persistent disks. ??? discusses the practice of stopping and starting Hadoop clusters in more detail.

---

<sup>1</sup> The `/tmp` directory may map to persistent storage, but can still be cleared out by the operating system on reboot.



```

<property>
  <name>fs.defaultFS</name>
  <value>hdfs://203.0.113.101:8020</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/${user.name}/tmp</value>
</property>

```

For Azure only, edit `/etc/hadoop/hdfs-site.xml` and disable the requirement that datanodes have a resolvable IP address. Azure does not establish reverse DNS lookup for private IP addresses by default.

```

<property>
  <name>dfs.namenode.datanode.registration.ip-hostname-check</name>
  <value>>false</value>
</property>

```

Edit `/etc/hadoop/yarn-site.xml` and make the following changes.

- Configure the `yarn.resourcemanager.hostname` property for your manager instance. Use its private IP address here as well.
- Configure the `yarn.nodemanager.aux-services` property with the value “map-reduce\_shuffle”. This enables YARN node managers to perform the shuffle stage of MapReduce jobs.

```

<property>
  <description>The hostname of the RM.</description>
  <name>yarn.resourcemanager.hostname</name>
  <value>203.0.113.101</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

```

Create the file `/etc/hadoop/mapred-site.xml`; the easiest way is to copy the empty template file for it.

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

Then, add the `mapreduce.framework.name` property with the value “yarn” so that the cluster will use YARN for executing MapReduce jobs. Also set the `yarn.app.mapreduce.am.staging-dir` property to the `/user` directory in HDFS; this will allow users running MapReduce jobs to write staging data into their home directories in HDFS, which will have the necessary permissions.

```

<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>

```

```
<property>
  <name>yarn.app.mapreduce.am.staging-dir</name>
  <value>/user</value>
</property>
```

## Finishing up Configuration

Edit `/etc/hadoop/slaves` and replace its contents with a list of the private IP addresses for the worker instances. This only needs to be done on the manager instance. This file is used by Hadoop's helper scripts to find each worker.

## Startup

The cluster is ready to start. Begin by formatting HDFS as usual. Log in to the manager instance as the “hdfs” user for the step.

```
# as hdfs
$ hdfs namenode -format mycluster
```

Now start up HDFS and YARN using the helper scripts, also on the manager instance. This uses the passwordless SSH set up earlier to connect to the manager itself and each worker to start the daemons. If you did not try out the connections manually and accept the key fingerprints for each instance, then you will see the interactive prompts emitted by these scripts; just type “yes” for each one you see and they should continue executing.

```
# as hdfs
$ $HADOOP_PREFIX/sbin/start-dfs.sh
# as yarn
$ $HADOOP_PREFIX/sbin/start-yarn.sh
```

*Example 5-1.*

For brevity, configuration and startup of the MapReduce Job History Server is omitted; the cluster will function without it. To set it up, establish a “mapred” user in the “hadoop” group, just like the “hdfs” and “yarn” users, and follow the standard instructions to configure and start the server.

## SSH Tunneling

Now that all of the daemons are started, you can check their statuses through their web interfaces. However, connecting directly will not work, because the security rules governing the instances block access from outside to the necessary ports. You can open up access to those ports from your IP address by adding new rules, but a better option is to use SSH tunneling, which maps a local port on your computer to a remote port on another computer through SSH. Using SSH tunnels lets you leave

only the SSH port open to anything outside the security group, and encrypts the traffic as a bonus.

You can create SSH tunnels using the standard SSH client. Run commands like these to establish tunnels from the local ports 50070 (for the namenode web interface) and 8088 (for the resource manager web interface) to the same ports on your manager instance. Use the public IP address or, if it is available, the public DNS name for your manager instance to establish the connection from your local computer. For the remote end of the tunnel, “localhost” works for the namenode, but you must use the same IP address for the resource manager as configured for the `yarn.resourcemanager.hostname` configuration property in `yarn-site.xml`.

Consult the man page for `ssh` for more information about setting up SSH tunnels. Tunneling is also covered in more detail in ???.

```
$ ssh -i /path/to/cloud_provider_key.pem -n -N \  
> -L 50070:localhost:50070 userid@manager.cloud-provider.example &  
$ ssh -i /path/to/cloud_provider_key.pem -n -N \  
> -L 8088:203.0.113.101:8088 userid@manager.cloud-provider.example &
```

Now, in your browser, navigate to <http://localhost:50070> and <http://localhost:8088> to see your namenode and resource manager ready for work. The web interfaces are being served from the manager instance, but tunneled through SSH to your browser.

## Running a Test Job

After all of that work, it should be gratifying to see your cluster in action. The Hadoop distribution comes with many example programs bundled in an examples JAR. You can use any of those on your new cluster. One easy one to try is the “pi” example, which calculates the value of pi using a MapReduce algorithm.

MapReduce jobs in the cluster will write their history information to `/user/history` in HDFS, so that directory needs to be established before any jobs are run. Using the “hdfs” user, create it with wide-open permissions but owned by the “mapred” user and the “hadoop” group.

```
# as hdfs  
$ hdfs dfs -mkdir -p /user/history  
$ hdfs dfs -chmod -R 1777 /user/history  
$ hdfs dfs -chown mapred:hadoop /user/history
```

Now, you can run the example under any account on an instance. For the account you choose, create a home directory in HDFS, using the hdfs account. Then go ahead and run the example.

```
# as hdfs  
$ hdfs dfs -mkdir -p /user/userid  
$ hdfs dfs -chown userid /user/userid
```

```
# as the user account
$ export PATH=/opt/hadoop/bin:$PATH
$ export HADOOP_CONF_DIR=/etc/hadoop
$ hadoop jar \
> /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-x.y.z.jar \
> pi 10 100
```

You should see the MapReduce job submitted to YARN and visible in the resource manager web interface. If not, then it is possible that you are not configured to use YARN for MapReduce, and the job is being run locally; check over the configuration, especially *yarn-site.xml* and *mapred-site.xml*.

After a short run time and some output describing the progress of the job, you should see an answer like this. Yours may vary.

```
Estimated value of Pi is 3.148000000000000000000000
```

Congratulations, you have created a functional Hadoop cluster in the cloud! You should feel free to try running more example programs or other analyses through it. Even though it is running in the cloud, using it is much like using an on-premises cluster.

## What If the Job Hangs?

If the example job never gets started, but gets stuck waiting for an application master (AM) to be allocated, it's usually a sign that there may not be enough memory on the node managers for allocating the container needed for the application master process. The default configurations for YARN are higher than what you often have to work with in smaller cloud deployments. Try setting these configuration properties:

- In */etc/hadoop/yarn-site.xml*, set `yarn.nodemanager.resource.memory-mb` to something low, for example, 1024 for a worker instance with 8 GB of memory. The default value is 8192 is too high for such an instance.
- In */etc/hadoop/mapred-site.xml*, set `yarn.app.mapreduce.am.resource.mb` to the same value.

If this does not work, consult documentation about tuning YARN configurations for MapReduce. They provide advice, and even worksheets, for coming up with appropriate configuration property settings for your cluster.

## Running Basic Data Loading and Analysis

The example job does exercise your new cluster, but might not be representative of the type of work a cloud-based cluster will perform. The next test of the cluster will get a little closer to that by running a couple of simple MapReduce jobs that read in

text data from Wikipedia and then, following the tradition of new Hadoop clusters everywhere, count words in it.

## Wikipedia Exports

Wikipedia offers on-demand and pre-generated exports of all of its data in XML format. You can export the complete information for a single article, or for a category of articles, on demand. Wikipedia also periodically posts large, complete dumps of many wikis in the Wikimedia family. These exports are well suited for analysis in a Hadoop cluster.

Visit <https://dumps.wikimedia.org/> to see the available pre-generated dumps.

## Analyzing a Small Export

While it is tempting to grab a complete Wikipedia dump, it is better to start with a smaller one, and then scale up. Since the XML format is the same for any dump, jobs will work on any of them.

### Generating the Export

Visit <https://en.wikipedia.org/wiki/Special:Export>, which lets you export one or more articles from the English Wikipedia site. Collect a set of page titles that you want to export and enter them manually, or select a category. In the example below, the category is “Member states of the United Nations”, which returns around two hundred articles. Remove any subcategory entries, and be sure to check the boxes to include only the current revision of each article and to save as a file. Then press the “Export” button to receive the exported data as XML.

Add pages from category:

Member states of the United Nations Add

Add pages manually:

Andorra  
Bahrain  
Category:United Nations Security Council resolutions concerning United Nations membership  
Cameroon  
Georgia\_(country)  
Grenada  
Guinea-Bissau  
Jamaica  
Kiribati  
Lesotho  
Malawi

☒ Include only the current revision, not the full history

☐ Include templates

☒ Save as file

Export

Figure 5-1. Wikipedia Export page

The export will serve as the source data for analysis, so it needs to be copied into HDFS in the cluster. Copy the file up to the manager instance using `scp`; be sure to use the same private key that works for SSH access. Then, on the manager instance, use the `hdfs` utility to copy the XML into the home directory for the same user account used earlier to run the `pi` example.

```
# on the computer where the export was downloaded
$ scp -i /path/to/cloud_provider_key.pem Wikipedia-20160701000000.xml \
> userid@manager.cloud-provider.example:.
# on the manager instance
$ hdfs dfs -copyFromLocal Wikipedia-20160701000000.xml /user/userid
```

## The MapReduce Jobs

For this analysis, two MapReduce jobs will be used. The first one will scan the XML and extract out each article's title and page contents, saving them into HDFS; this can be seen as a very basic *extract-transform-load* or *ETL* process, although real-life ETL is often much more involved. The second one will evaluate each article's text to count words in it, much like the standard Hadoop word count example.

Here are the important parts of the Java source<sup>2</sup> for the loader job, which consists of a driver class and a mapper class. This is a map-only job with no reducers required. The driver sets up the job to stream the XML data from the file provided as the first argument, run it through a mapper, and send the output data to sequence files in an HDFS directory specified by the second argument.

*Example 5-2. Wikipedia loader job driver code*

```
JobConf conf = new JobConf(getClass());
conf.setJobName("WP dump loader");

// Set the mapper class, but skip the reduce phase
conf.setMapperClass(WikipediaDumpLoaderMapper.class);
conf.setNumReduceTasks(0);
// The object key/value pairs are text
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(Text.class);

// Stream XML into the job
conf.setInputFormat(StreamInputFormat.class);
StreamInputFormat.addInputPath(conf, new Path(args[0]));
// Use the XML record reader, with each page as one record
conf.set("stream.recordreader.class",
        "org.apache.hadoop.streaming.StreamXmlRecordReader");
conf.set("stream.recordreader.begin", "<page>");
conf.set("stream.recordreader.end", "</page>");
// Emit sequence files
conf.setOutputFormat(SequenceFileOutputFormat.class);
SequenceFileOutputFormat.setOutputPath(conf, new Path(args[1]));

JobClient.runJob(conf);
return 0;
```

*Example 5-3. Wikipedia loader job mapper code*

```
private enum Counter { ARTICLES }

private DocumentBuilder db;

@Override
public void configure(JobConf conf) {
    try {
        db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    } catch (ParserConfigurationException e) {
        throw new IllegalStateException("XML parser configuration is bad", e);
    }
}
```

---

<sup>2</sup> Find the complete source at: <https://github.com/bhavanki/hadoop-in-the-cloud>

```

}

@Override
public void map(Text key, Text value, OutputCollector<Text, Text> output,
                Reporter reporter) throws IOException {
    try {
        // Parse the page of XML into a document
        Document doc = db.parse(new InputSource(new StringReader(key.toString())));

        // Extract the title and text (article content) from the page content
        String title = doc.getElementsByTagName("title").item(0).getTextContent();
        String text = doc.getElementsByTagName("text").item(0).getTextContent();

        // Emit the title and text pair
        output.collect(new Text(title), new Text(text));
        reporter.getCounter(Counter.ARTICLES).increment(1L);
    } catch (SAXException e) {
        throw new IOException(e);
    }
}
}

```

MapReduce veterans may notice that this job uses the old Java MapReduce API. This is because `StreamInputFormat` is currently only available under that API. An alternative design would use Hadoop Streaming to load the XML data.

The `StreamInputFormat` is configured to simply send each article’s content to the mapper as the key; the value sent to the mapper is always an empty string. Other than those details, the job is straightforward.

Once these classes are packaged into a JAR, the job is ready to run. Using `scp`, copy the JAR to the manager instance under your user account, and run it using the `yarn` command line utility.

```

$ yarn jar basic-loader-1.0.0-SNAPSHOT.jar com.dahitc.WikipediaDumpLoaderDriver \
> Wikipedia-20160701000000.xml wikitext

```

As before, you will see signs of progress as the job is run on the cluster. When the job completes, there will be one or more sequence files in a directory in HDFS (the example above uses the *wikitext* directory in the user’s HDFS home directory). You should also see the custom “ARTICLES” counter reporting the total number of articles parsed. If you like, you can peek at a sequence file to check that article text is present; while a sequence file is binary, these will have large areas of plain text that are easy to discriminate.

```

$ hdfs dfs -ls wikitext
$ hdfs dfs -cat wikitext/part-00000

```

It’s time for the second job, also written in Java, which performs the word count. This job works like any other word count job, except that it reads from sequence files instead of ordinary text files.



#### Example 5-4. Wikipedia word count job driver code

```
JobConf conf = new JobConf(getClass());
conf.setJobName("WP word count");

// Set the mapper and reducer classes, and use the reducer as a combiner
conf.setMapperClass(WikipediaWordCountMapper.class);
conf.setReducerClass(WikipediaWordCountReducer.class);
conf.setCombinerClass(WikipediaWordCountReducer.class);
// The object key/value pairs are text words and integer counts
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);

// Read in sequence files
conf.setInputFormat(SequenceFileInputFormat.class);
SequenceFileInputFormat.addInputPath(conf, new Path(args[0]));
// Emit ordinary text files
conf.setOutputFormat(TextOutputFormat.class);
TextOutputFormat.setOutputPath(conf, new Path(args[1]));

JobClient.runJob(conf);
return 0;
```

#### Example 5-5. Wikipedia word count job mapper code

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

@Override
public void map(Text key, Text value, OutputCollector<Text, IntWritable> output,
                Reporter reporter) throws IOException {
    // Split the text content of the article on whitespace
    String[] words = value.toString().split("\\s+");
    // Count each word occurrence
    for (String word : words) {
        wordText.set(word);
        output.collect(wordText, ONE);
    }
}
```

#### Example 5-6. Wikipedia word count job reducer code

```
private IntWritable sumIntWritable = new IntWritable();

/**
 * key = word
 * values = counts
 */
@Override
public void reduce(Text key, Iterator<IntWritable> values,
                  OutputCollector<Text, IntWritable> output, Reporter reporter)
```

```

throws IOException {

    // Total up the incoming counts for the word
    int sum = 0;
    while (values.hasNext()) {
        sum += values.next().get();
    }
    // Emit the word count
    sumIntWritable.set(sum);
    output.collect(key, sumIntWritable);
}

```

The old Java MapReduce API is used here, for consistency with the loader job, but this code could easily be rewritten to use the new API instead.

```

$ yarn jar basic-loader-1.0.0-SNAPSHOT.jar com.dahitc.WikipediaWordCountDriver \
> wikitext wikiwordcount

```

The results are in one or more files in HDFS under the *wikiwordcount* directory. The mapper code used here is intentionally simple; differing letter case and punctuation are not accounted for, so the results could be cleaned up with some improvements.

```

$ hdfs dfs -ls wikiwordcount
$ hdfs dfs -cat wikiwordcount/part-00000

```

Both of these MapReduce jobs are quite simple, and could have been combined into a single job, but they make simpler examples when separate. Also, the results of the first job could be saved off and used as the starting point for other analyses later, saving the work of parsing XML in each run. In a cloud architecture, you have the choice of saving valuable intermediate data long term in object storage services (see ???), which lets you trade off the cost of repeated computation for the cost of increased data storage. As computation time increases, the option of reducing it by cheaply storing useful data products becomes more attractive.

## Go Bigger

With the Wikipedia export jobs available and working in your cluster, you have a starting point for further exploration. The example above used an export of around two hundred articles, which at the time of writing weighed in at 26 MB, but larger exports should work just as well. The only limitation is the amount of room available in HDFS, since the XML data needs to be loaded there first for the loader job to work on it. If you are low on room in HDFS, or find that your workers run out of disk space when spilling data, either add worker instances or increase the available block storage for the existing workers<sup>3</sup>.

---

<sup>3</sup> Increasing volume storage can be easy or difficult, depending on the cloud provider.

The trickiest part of working with a large export is getting it from Wikipedia into HDFS. In this chapter, the file was downloaded to the local computer, copied to the manager instance, and then copied into HDFS. This approach may not work well for, say, the complete export of English language Wikipedia, which exceeds 20 GB, compressed, as of this writing.

It is feasible to download a large file directly to an instance, and then copy the file from there into HDFS. The disk where the large file resides can be kept as the “golden copy” of the data, and attached to instances when the file stored on it is needed.

A more advanced approach involves uploading a large file to the cloud provider’s object storage service, and then configuring Hadoop to access the file from there. The exact mechanics of this approach vary with the cloud provider, and there can be multiple ways to pull it off. Some possibilities:

- Under AWS: Download the file to an EC2 instance, decompress it there, and use the S3 command-line client to upload the file into S3. Then, use Hadoop’s `s3a` protocol and the `distcp` tool to copy the file from S3 into HDFS.
- Under GCP, use the Google Cloud Storage Transfer Service to load the file directly from Wikipedia into a bucket in Cloud Storage<sup>4</sup>. Then, install the Google Cloud Storage Connector into your cluster and access the file directly as an HDFS volume.
- Under Azure, download the file to a virtual machine, decompress it there, and use the Azure CLI to upload it to Blob Storage. Then, install the Azure Blob Storage module into your cluster and access the file directly as an HDFS volume.

???, which covers adding Apache Hive to your cluster, describes how to use the `s3a` filesystem connector to directly access data residing in S3 for use in Hive queries.

Check your cloud provider’s documentation and examples to see what options you have for working with files and the provider’s storage features. As new features are introduced, you may discover better patterns to move your large files around.

---

<sup>4</sup> With the correct content type set for the file, Cloud Storage should decompress it when it is accessed