Big Data Architect's Handbook

A guide to building proficiency in tools and systems used by leading big data experts

Packt

www.packt.com

By Engr. Syed Muhammad Fahad Akhtar

Big Data Architect's Handbook

A guide to building proficiency in tools and systems used by leading big data experts

Syed Muhammad Fahad Akhtar

Packt

BIRMINGHAM - MUMBAI

Big Data Architect's Handbook

Commissioning Editor: Sunith Shetty
Acquisition Editor: Namrata Patil
Content Development Editor: Aaryaman Singh
Technical Editor: Dinesh Chaudhary
Copy Editor: Safis Editing
Project Coordinator: Manthan Patel
Proofreader: Safis Editing
Indexer: Mariammal Chettiyar
Graphics: Tania Dutta
Production Coordinator: Deepika Naik

First published: June 2018

Production reference: 1190618

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78883-582-4

`www.packtpub.com`

Contributors

About the author

Syed Muhammad Fahad Akhtar has 12+ years of industry experience in analysis, designing, developing, integrating, and managing large applications in different and diverse industries. He has vast exposure of working in UAE, Pakistan, and Malaysia. He is currently working in ASIT Solutions as a solutions architect in Malaysia.
He received his master's degree from Torrens University, Australia, and bachelor of science in computer engineering from National University of Computer and Emerging Sciences (FAST), Pakistan.

He has cross platform expertise and achieves recognition from IBM, Sun Microsystems and Microsoft. Fahad has received the following accolades:

- IBM Certified Big Data Architect - 2017
- Sun Certified Java Programmer - 2012
- Microsoft Certified Solution Developer - 2009
- Microsoft Certified Application Developer - 2007
- Microsoft Certified Professional - 2005

He also contributed his experience and services towards as Member, board of director in K.K. Abdal Institute of Engineering and Management Sciences, Pakistan and is a board member of Alam Educational Society.

You can find him on LinkedIn at `syedmfahad`.

About the reviewers

Albenzo Coletta is a senior software and systems engineer in robotic, defense, avionic, telecommunication fields. He has a master's in computational robotics models. He was an industrial researcher in AI. He was also a designer for a robotic communication system for COMAU, as a business analyst. He designed a Neuro Fuzzy system for financial problems (with Sannio University) and recommender system for major Italian Editorial groups. He was a consultant at UCID (Economic and Financial Ministry), He also made a Mobile Human Robotic Interaction System.

Giancarlo Zaccone has 10+ years of experience in managing research projects in both scientific and industrial areas. He was a researcher at CNR, the National Research Council of Italy, where he was involved in projects on parallel numerical computing and scientific visualization. He is a senior software engineer at a consulting company, developing and testing software systems for space and defense applications.
He holds a master's in physics from Federico II of Naples and a 2nd-level postgraduate master course in scientific computing from La Sapienza of Rome.

Thirukkumaran Haridass is an independent IT consultant based out of Chennai, India. He works with clients from various verticals to help them build big data and AI solutions for their business needs. He is also the author of Learning Google BigQuery published by Packt. Prior to becoming a consultant, he worked in USA for over 13 years and 6 years at Builder Homesite Inc, Austin, Texas, USA. He also worked for Fortune 500 companies such as JCPenney and Volkswagen as a consultant.

Table of Contents

Preface

Big data architects are the masters of data and hold high value in today's market. Handling big data, be it of good or bad quality, is not an easy task. The prime task before any big data architect is to build an end-to-end big data solution that integrates data from different sources and analyzes it to find useful, hidden insights.
Big Data Architect's Handbook takes you through developing a complete, end-to-end big data pipeline that will lay the foundation for you and provide the necessary knowledge required to be an architect in big data. Right from understanding the design considerations to implementing a solid, efficient, and scalable data pipeline, this book walks you through all the essential aspects of big data. It also gives you an overview of how you can leverage the power of various big data tools such as Apache Hadoop and Elasticsearch in order to bring them together and build an efficient big data solution.
By the end of this book, you will be able to build your own design system that integrates, maintains, visualizes, and monitors your data. In addition, you will have a smooth design flow in each process, putting insights in action.

Who this book is for

Big Data Architect's Handbook is for you if you are an aspiring data professional, developer, or IT enthusiast who aims to be an all-round architect in big data. This book is a one-stop solution to enhance your knowledge and carry out easy to complex activities required to become a big data architect.

What this book covers

`Chapter 1`, *Why Big Data?*, explains what big data is, why we need big data, who should deal with big data, when to use big data, and how to use big data. The design consideration of the end-to-end big data solution, including cloud, Hadoop, network, analytics and so on, are also outlined here.

`Chapter 2`, *Big Data Environment Setup*, provides a step-by-step guide of how to setup environment to run big data applications.

`Chapter 3`, *Hadoop Ecosystem*, is about the Hadoop ecosystem. It consists of different open source modules, accessories, and Apache projects for reliable and scalable distributed computing. This chapter will teach you how to build a Hadoop big data system for streaming data with a step-by-step guide.

`Chapter 4`, *NoSQL Database*, explains the concepts, principles, properties, performance and hybrid of the popular NoSQL database so that a big data architect can confidently choose appropriate NoSQL for their projects. This chapter will teach you how to implement NoSQL for killer applications with a step-by-step guide.

`Chapter 5`, *Off-the-Shelf Commercial Tools*, introduces some popular commercial off-the-shelf tools for big data with a hands-on Stream Analytics example.

`Chapter 6`, *Containerization*, introduces the concept and application of container-based virtualization. It is an OS-level virtualization method for deploying and running distributed applications without launching an entire VM for each application. Moreover, management of Dockers and Kubernetes using Openshift is demonstrated here.

`Chapter 7`, *Network Infrastructure*, teaches essential network technology for an architect to design big data systems across racks, data centers, and geographical locations. Moreover, this chapter will teach you the network visualization tool via a step-by-step guide.

`Chapter 8`, *Cloud Infrastructure*, introduces essential considerations on cloud infrastructure design for big data from the perspective of performance and capability. The requirements of deploying big data in cloud are unique and quite different from traditional applications. Therefore, a big data architect must need careful design, especially estimating the amount of data to analyze by using the big data capability in the cloud, because not all public or private cloud offerings are built to accommodate big data solutions.

`Chapter 9`, *Security and Monitoring*, is about essential knowledge on security, including next-generation firewalls, DevOps security, and monitoring tools.

`Chapter 10`, *Frontend Architecture*, introduces the Frontend architecture, which is a collection of tools and processes that aims to improve the quality of our frontend code while creating a more efficient, scalable, and sustainable design for big data systems. To be a successful big data Architect, one critical factor is to present persuasive analytic results to mostly non-technical persons, such as C-level management, and decision-makers with a user-friendly, elegant, and responsive user graphic interface. This chapter will teach you how to use the React + Redux framework to build a responsive and easy debug user interface.

`Chapter 11`, *Backend Architecture*, shows how to design a scalable, resilient, manageable, and cost-effective distributed backend architecture with different combinations of technology. It handles business logic and data storage with a RESTful web API service.

`Chapter 12`, *Machine Learning*, teaches the essential concepts and killer applications of Machine Learning. You will learn about the most effective machine learning techniques, and gain practice implementing them and getting them to work for yourself and your enterprise. You'll learn about not only the theoretical underpinnings of learning, but also the practical know-how needed to quickly and powerfully apply these techniques to new problems.

`Chapter 13`, *Artificial Intelligence*, introduces AI and CNN with hands-on big data killer applications. The application for CNN or deep learning to work with machine learning is one good method to handle unstructured big data.

`Chapter 14`, *Elasticsearch*, shows how to use the open source tool Elasticsearch to do searching tasks in a big data system. This is because it is an enterprise-grade search engine, and easy to scale. More features of it are: handy REST API and JSON response, good documentation, Sense UI, stable and proven Lucene underlying engine, excellent Query DSL, multi-tenancy, advanced Search Features, configurable and extensible, percolation, custom analyzer, On-the-Fly Analyzer selection, rich ecosystem, and active community.

`Chapter 15`, *Structured Data*, introduces the use of open source tools to manipulate and analyze structured data.

`Chapter 16`, *Unstructured Data*, shows how to use open source tools to manipulate and analyze unstructured data. The readers will learn how to use machine learning and AI to extract information for analysis in killer applications such as a Retail Recommendation System and Facial Recognition.

`Chapter 17`, *Data Visualization*, illustrates how to use tools to present analytical results to users using two top-of-the-shelf tools, Matplotlib and D3.js.

`Chapter 18`, *Financial Trading System*, covers algorithmic trading benefits and strategies, and how to design and deploy an end-to-end Financial Trading System via a step-by-step guide.

`Chapter 19`, *Retail Recommendation System*, shows how to design and deploy an end-to-end Retail Recommendation System through a step-by-step guide.

To get the most out of this book

1. This book uses Ubuntu Linux desktop environment to setup and execute all the example and sample codes.
2. Each chapter contains the installation and setup instruction of the framework / application used. Follow those instructions carefully in order to setup the environment and successfully run the provided example.

Download the example code files

You can download the example code files for this book from your account at `www.packtpub.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packtpub.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Big-Data-Architects-Handbook`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `http://www.packtpub.com/sites/default/files/downloads/BigDataArchitectsHandbook_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Mount the downloaded `WebStorm-10*.dmg` disk image file as another disk in your system."

A block of code is set as follows:

```
html, body, #map {
 height: 100%;
 margin: 0;
 padding: 0
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Any command-line input or output is written as follows:

```
$ mkdir css
$ cd css
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Select **System info** from the **Administration** panel."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email feedback@packtpub.com and mention the book title in the subject of your message. If you have questions about any aspect of this book, please email us at questions@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packtpub.com.

1
Why Big Data?

Nowadays, it seems like everyone is talking about the term big data. However, a majority of them are not sure what it is and how they will make the most out of it. Apart from a few companies, most of them are still confused about the concept and are not ready to adopt the idea. Even when we hear the term big data, so many questions come to our minds. It is very important to understand these concepts. These questions include:

- What is big data?
- Why is there so much hype about it?
- Does it just mean huge volumes of data or is there something else to it?
- Does big data have any characteristics and what are these?
- Why do we need big data architects and what are the design considerations we have to consider to architect any big data solutions?

In this chapter, we will focus on answering these questions and building a strong foundation toward understanding the world of big data world. Mainly, we will be covering the following topics:

- Big data
- The characteristics of big data
- The different design considerations to big data solutions
- The key terminology used in the world of big data

Let's now start with the first and the most important question: what is big data?

What is big data?

If we take a simpler definition, it can basically be stated as a huge volume of data that cannot be stored and processed using the traditional approach. As this data may contain valuable information, it needs to be processed in a short span of time. This valuable information can be used to make predictive analyses, as well as for marketing and many other purposes. If we use the traditional approach, we will not be able to accomplish this task within the given time frame, as the storage and processing capacity would not be sufficient for these types of tasks.

That was a simpler definition in order to understand the concept of big data. The more precise version is as follows:

Data that is massive in volume, with respect to the processing system, with a variety of structured and unstructured data containing different data patterns to be analyzed.

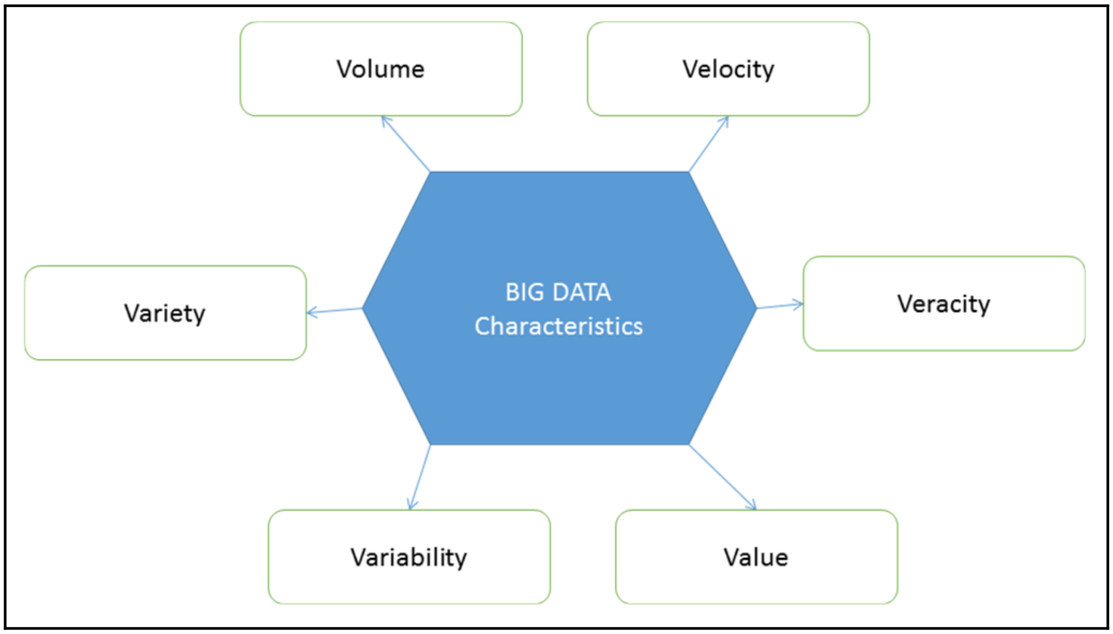
From traffic patterns and music downloads, to web history and medical records, data is recorded, stored, and analyzed to enable technology and services to produce the meaningful output that the world relies on every day. If we just keep holding on to the data without processing it, or if we don't store the data, considering it of no value, this may be to the company's disadvantage.

Have you ever considered how YouTube is suggesting to you the videos that you are most likely to watch? How Google is serving you localized ads, specifically targeted to you as ones that you are going to open, or of the product you are looking for? These companies are keeping all of the activities you do on their website and utilizing them for an overall better user experience, as well as for their benefit, to generate revenue. There are many examples available of this type of behavior and it is increasing as more and more companies are realizing the power of data. This raises a challenge for technology researchers: coming up with more robust and efficient solutions that can cater to new challenges and requirements.

Now, as we have some understanding of what big data is, we will move ahead and discuss its different characteristics.

Characteristics of big data

These are also known as the dimensions of big data. In 2001, Doug Laney first presented what became known as the **three Vs of big data** to describe some of the characteristics that make big data different from other data processing. These three Vs are volume, velocity, and variety. This the era of technological advancement and loads of research is going on. As a result of this reaches and advancements, these three Vs have become the **six Vs of big data** as of now. It may also increase in future. As of now, the six Vs of big data are volume, velocity, variety, veracity, variability, and value, as illustrated in the following diagram. These characteristics will be discussed in detailed later in the chapter:



Characteristics of big data

Different computer memory sizes are listed in the following table to give you an idea of the conversions between different units. It will let you understand the size of the data in upcoming examples in this book:

1 Bit	**Binary digit**
8 Bits	1 byte
1,024 Bytes	1 KB (kilobyte)
1,024 KB	1 MB (megabyte)

1,024 MB	1 GB (gigabyte)
1,024 GB	1 TB (terabyte)
1,024 TB	1 PB (petabyte)
1,024 PB	1 EB (exabyte)
1,024 EB	1 ZB (zettabyte)
1,024 ZB	1 YB (yottabyte)
1,024 YB	1 brontobyte
1,024 brontobyte	1 geopbyte

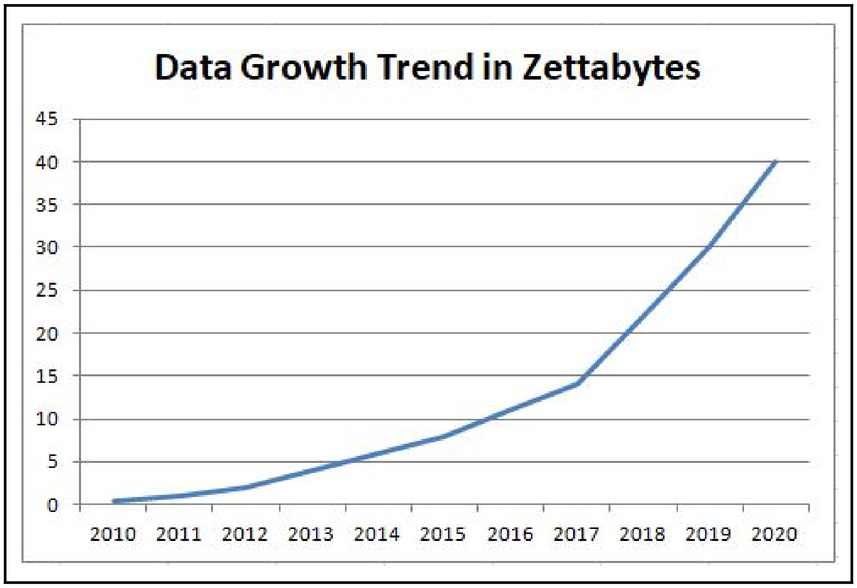
Now that we have established our basis for subsequent discussions, let's move on to discuss the first characteristics of big data.

Volume

In earlier years, company data only referred to the data created by their employees. Now, as the use of technology increases, it is not only data created by employees but also the data generated by machines used by the companies and their customers. Additionally, with the evolution of social media and other internet resources, people are posting and uploading so much content, videos, photos, tweets, and so on. Just imagine; the world's population is 7 billion, and almost 6 billion of them have cell phones. A cell phone itself contains many sensors, such as a gyro-meter, which generates data for each event, which is now being collected and analyzed.

When we talk about volume in a big data context, it is an amount of data that is massive with respect to the processing system that cannot be gathered, stored, and processed using traditional approaches. It is data at rest that is already collected and streaming data that is continuously being generated.

Take the example of Facebook. They have 2 billion active users who are continuously using this social networking site to share their statuses, photos, videos, commenting on each other's posts, likes, dislikes, and many more activities. As per the statistics provided by Facebook, a daily 600 TB of data is being ingested into the database of Facebook. The following graph represents the data that was there in previous years, the current situation and where it is headed in future:



Past, present and future data growth

Take another example of a jet airplane. One statistic shows that it generates 10 TB of data for every hour of flight time. Now imagine, with thousands of flights each day, how the amount of data generated may reach many petabytes every day.

In the last two years, the amount of data generated is equal to 90% of the data ever created. The world's data is doubling every 1.2 years. One survey states that 40 zettabytes of data will be created by 2020.

Not so long ago, the generation of such massive amount of data was considered to be a problem as the storage cost was very high. But now, as the storage cost is decreasing, it is no longer a problem. Also, solutions such as Hadoop and different algorithms that help in ingesting and processing this massive amount of data make it even appear resourceful.

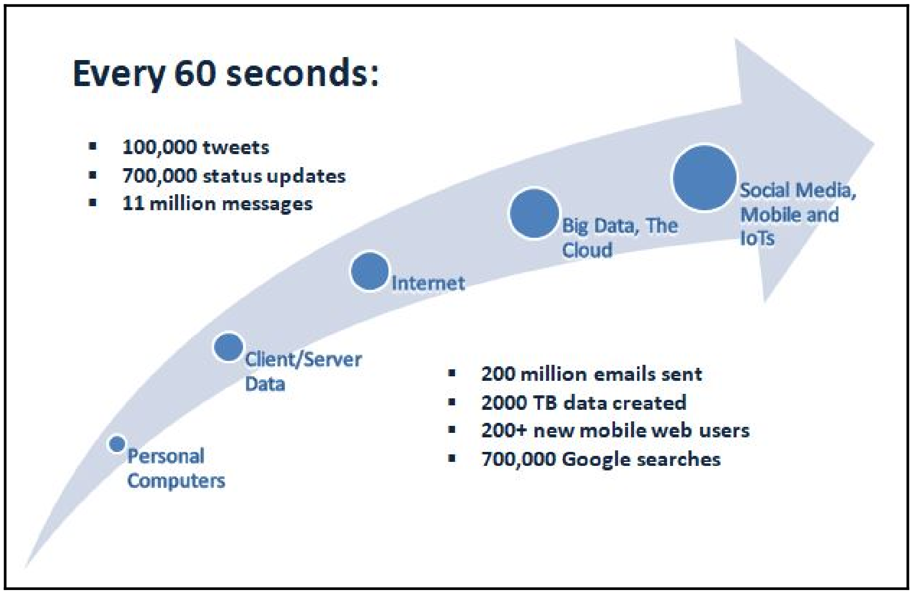
The second characteristic of big data is velocity. Let's find out what this is.

Velocity

Velocity is the rate at which the data is being generated, or how fast the data is coming in. In simpler words, we can call it data in motion. Imagine the amount of data Facebook, YouTube, or any social networking site is receiving per day. They have to store it, process it, and somehow later be able to retrieve it. Here are a few examples of how quickly data is increasing:

- The New York stock exchange captures 1 TB of data during each trading session.
- 120 hours of videos are being uploaded to YouTube every minute.
- Data generated by modern cars; they have almost 100 sensors to monitor each item from fuel and tire pressure to surrounding obstacles.
- 200 million emails are sent every minute.

If we take the example of social media trends, more data means more revealing information about groups of people in different territories:



Velocity at which the data is being generated

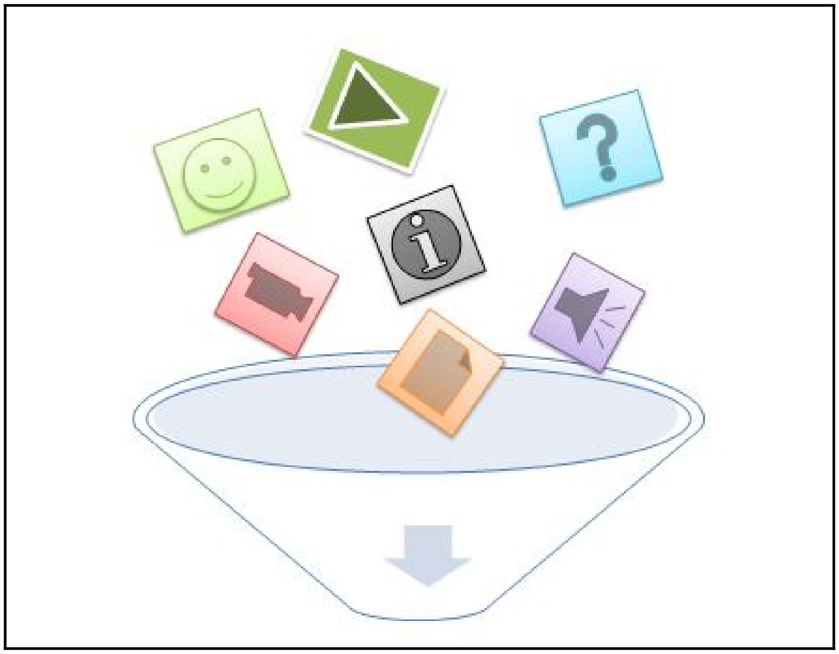
The preceding chart shows the amount of time users are spending on the popular social networking websites. Imagine the frequency of data being generated based on these user activities. This is just a glimpse of what's happening out there.

Another dimension of velocity is the period of time during which data will make sense and be valuable. Will it age and lose value over time, or will it be permanently valuable? This analysis is also very important because if the data ages and loses value over time, then maybe over time it will mislead you.

Till now, we have discussed two characteristics of big data. The third one is variety. Let's explore it now.

Variety

In this section, we study the classification of data. It can be structured or unstructured data. Structured data is preferred for information that has a predefined schema or that has a data model with predefined columns, data types, and so on, whereas unstructured data doesn't have any of these characteristics. These include a long list of data such, as documents, emails, social media text messages, videos, still images, audio, graphs, the output from all types of machine-generated data from sensors, devices, RFID tags, machine logs, and cell phone GPS signals, and more. We will learn more details about structured and unstructured data in separate chapters in this book:



Variety of data

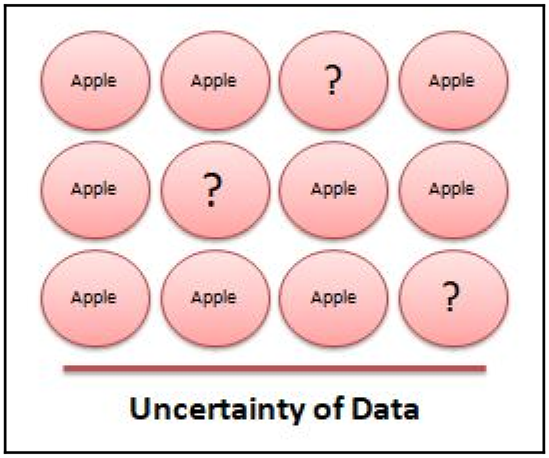
Let's take an example; 30 billion pieces of content are shared on Facebook each month. 400 million Tweets are sent per day. 4 billion hours of videos are watched on YouTube every month. These are all examples of unstructured data being generated that needs to be processed, either for a better user experience or to generate revenue for the companies itself.

The fourth characteristic of big data is veracity. It's time to find out all about it.

Veracity

This vector deals with the uncertainty of data. It may be because of poor data quality or because of the noise in data. It's human behavior that we don't trust the information provided. This is one of the reasons that one in three business leaders don't trust the information they use for making decisions.

We can consider in a way that velocity and variety are dependent on the clean data prior to analysis and making decisions, whereas veracity is the opposite to these characteristics as it is derived from the uncertainty of data. Let's take the example of apples, where you have to decide whether they are of good quality. Perhaps a few of them are average or below average quality. Once you start checking them in huge quantities, perhaps your decision will be based on the condition of the majority, and you will make an assumption regarding the rest, because if you start checking each and every apple, the remaining good-quality ones may lose their freshness. The following diagram is an illustration of the example of apples:



Veracity: illustration of uncertainty in apple example

The main challenge is that you don't get time to clean streaming data or high-velocity data to eliminate uncertainty. Data such as events data is generated by machines and sensors and if you wait to first clean and process it, that data might lose value. So you must process it as is, taking account of uncertainty.

Veracity is all about uncertainty and how much trust you have in your data, but when we use it in terms of the big data context, it may be that we have to redefine trusted data with a different definition. In my opinion, it is the way you are using data or analyzing it to make decisions. Because of the trust you have in your data, it influences the value and impact of the decisions you make.

Let's now look at the fifth characteristic of big data, which is variability.

Variability

This vector of big data derives from the lack of consistency or fixed patterns in data. It is different from variety. Let's take an example of a cake shop. It may have many different flavors. Now, if you take the same flavor every day, but you find it different in taste every time, this is variability. Consider the same for data; if the meaning and understanding of data keeps on changing, it will have a huge impact on your analysis and attempts to identify patterns.

Now comes the final and an important characteristic of big data—value.

Value

This is the most important vector in terms of big data, but is not particularly associated with big data, and it is equally true for small data as well. After addressing all the other Vs, volume, velocity, variety, variability, and veracity, which takes a lot of time, effort, and resources, now it's time to decide whether it's worth storing that data and investing in infrastructure, either on premises or in the cloud. One aspect of value is that you have to store a huge amount of data before you can utilize it in order to give valuable information in return. Previously, storing this volume of data lumbered you with huge costs, but now storage and retrieval technology is so much less expensive. You want to be sure that your organization gets value from the data. The analysis needs to be performed to meet ethical considerations.

Now that we have discussed and understand the six Vs of big data, it's time to broaden our scope of understanding and find out what to do with data having these characteristics. Companies may still think that their traditional systems are sufficient for data having these characteristics, but if they remain under this influence, they may lose in the long run. Now that we have understood the importance of data and its characteristics, the primary focus should be how to store it, how to process it, which data to store, how quickly an output is expected as a result of analysis and so on. Different solutions for handling this type of data, each with their own pros and cons, are available on the market, while new ones are continually being developed. As a big data architect, remember the following key points in your decision making that will eventually lead you to adopt one of them and leave the others.

Solution-based approach for data

Increasing revenue and profit is the key focus of any organization. Targeting this requires efficiency and effectiveness from employees, while minimizing the risks that affect overall growth. Every organization has a competitor and, in order to compete with them, you have to think and act quickly and effectively before your competitor does. Most decision-makers depend on the statistics made available to them and raise the following issues:

- What if you get analytic reports faster compared to traditional systems?
- What if you can predict how customers behave, different trends, and various opportunities to grow your business, in close to real time?
- What if you have automated systems that can initiate critical tasks automatically?
- What if automated activities clean your data and you can make your decisions based on reliable data?
- What if you can predict the risk and quantify them?

Any manager, if they can get the answers to these questions, can act effectively to increase the revenue and growth of any organization, but getting all these answers is just an ideal scenario.

Data – the most valuable asset

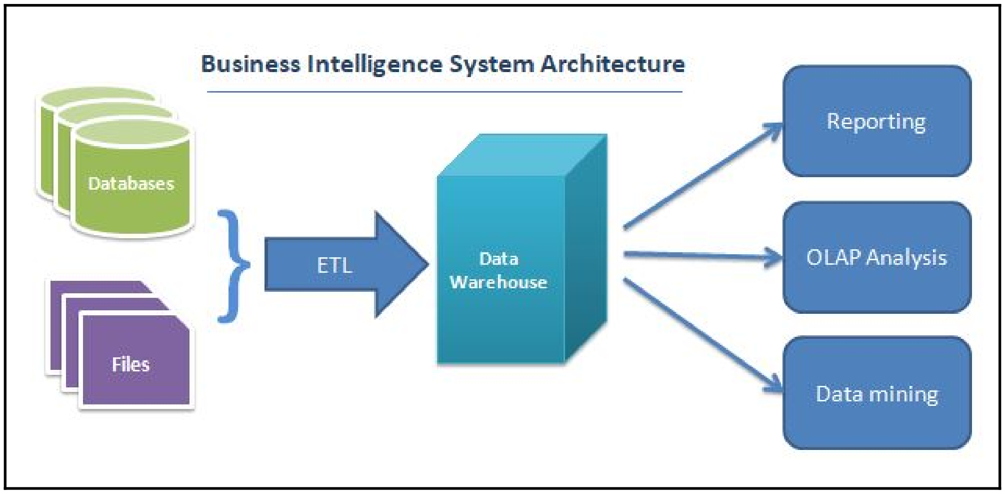
Almost a decade ago, people started realizing the power of data: how important it can be and how it can help organizations to grow. It can help them to improve their businesses based on the actual facts rather than their instincts. There were few sources of data to collect from, and analyzing this data in its raw form was not an easy task.

Now that the amount of data is increasing exponentially, at least doubling every year, big data solutions are required to make the most of your assets. Continuous research is being conducted to come up with new solutions and regular improvements are taking place in order to cater to requirements, following realization of the important fact that data is everything.

Traditional approaches to data storage

Human-intensive processes only work to make sense of data that doesn't scale as data volume increases. For example, people used to put each and every record of the company record in some sort of spreadsheet, and then it is very difficult to find and analyze that information once the volume or velocity of information increases.

Traditional systems use batch jobs, scheduling them on daily, weekly, or monthly bases to migrate data into separate servers or into data warehouses. This data has schema and is categorized as structured data. It will then go through the processing and analysis cycle to create datasets and extract meaningful information. These data warehouses are optimized for reporting and analytics purposes only. This is the core concept of **business intelligence (BI)** systems. BI systems store this data in relational database systems. The following diagram illustrates an architecture of a traditional BI system:



Business intelligence system architecture

The main issue with this approach is the latency rate. The reports made available for the decision makers are not real-time and are mostly days or weeks old, dependent on fewer sources having static data models.

This is an era of technological advancement and everything is moving very rapidly. The faster you respond to your customer, the happier your customer will be and it will help your business to grow. Nowadays, the source of information is not just your transactional databases or a few other data models; there are many other data sources that can affect your business directly and if you don't capture them and include them in your analysis, it will hit you hard. These sources include blog posts and web reviews, social media sources, posts, tweets, photos, videos, and audio. And it is not just these sources; logs generated by sensors, commonly known as **IoTs** (**Internet of Things**), in your smartphone, appliances, robot and autonomous systems, smart grids, and any devices that are collecting data, can now be utilized to study different behaviors and patterns, something that was unimaginable in the past.

It is a fact that every type of data is increasing, but especially IoTs, which generate logs of each and every event automatically and continuously. For example, a report shared by Intel states that an autonomous car generates and consumes 4 TB of data each day, and this is from just an hour of driving. This is just one source of information; if we consider all the previously mentioned sources, such as blogs and social media, it will not just make a few terabytes. Here, we are talking about exabytes, zettabytes, or even yottabytes of data.

We have talked about data that is increasing, but it is not just that; the types of data are also growing. Fewer than 20% of the types have definite schema. The other 80% is just raw data without any specific pattern which cannot reside in traditional relational database systems. This 80% of data includes videos, photos, and textual data including posts, articles, and emails.

Now, if we consider all these characteristics of data and try to implement this in a traditional BI solution, it will only be able to utilize 20% of your data, because the other 80% is just raw data, making it outreached for relational database systems. In today's world, people have realized that the data that they considered to be of no use can actually make a big difference to decision making and to understanding different behaviors. Traditional business solutions, however, are not the correct approach to analyze data with these characteristics, as they mostly work with definite schema and on batch jobs that produce results after a day, week or month.

Clustered computing

Before we take a further dive into big data, let us understand clustered computing. This is a set of computers connected to each other in such a way that they act as a single server to the end user. It can be configured to work with different characteristics that enable high availability, load balancing, and parallel processing. Each computer in these configurations is called a **node**. They work together to execute any task or application and behave as a single machine. The following diagram illustrates a computer cluster environment:

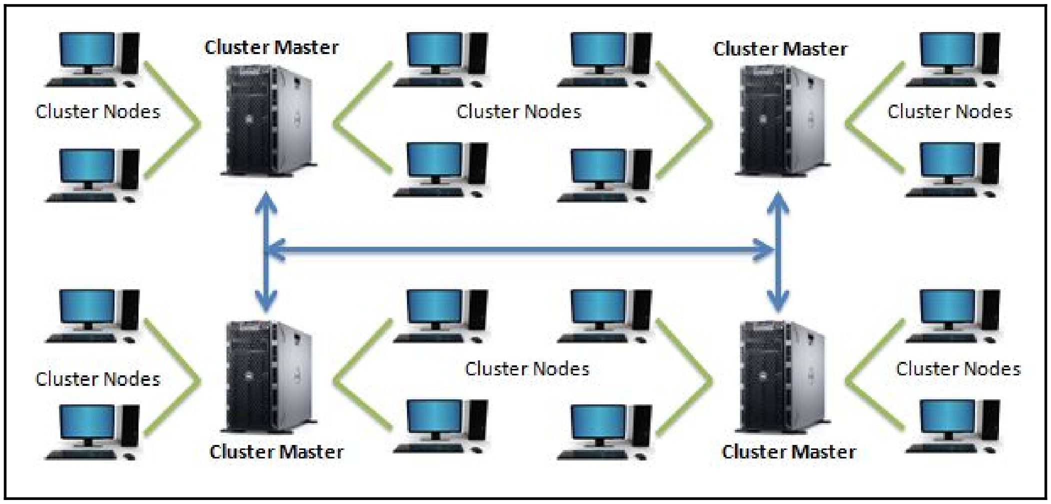


Illustration of a computer clustered environment

The volume of data is increasing, as we have already stated; it is now beyond the capabilities of a single computer to do the analysis all by itself. Clustered computing combines the resources of many smaller low — cost machines, to achieve many greater benefits. Some of them are listed here in the following sections

High availability

It is very important for all companies that their data and content must be available at all times and that when any hardware or software failure occurs, it must not constitute a disaster for them. Cluster computing provides fault tolerance tools and mechanisms to provide maximum uptime without affecting performance, so that everyone has their data ready for analysis and processing.

Resource pooling

In clustered computing, multiple computers are connected to each other to act as a single computer. It is not just that their data storage capacity is shared; CPU and memory pooling can also be utilized in individual computers to process different tasks independently and then merge outputs to produce a result. To execute large datasets, this setup provides more efficient processing power.

Easy scalability

Scaling is very straightforward in clustered computing. To add additional storage capacity or computational power, just add new machines with the required hardware to the group. It will start utilizing additional resources with minimum setup, with no need to physically expand the resources in any of the existing machines.

Big data – how does it make a difference?

We have established an understanding regarding traditional systems, such as BI, how they work, what their focused areas are, and where they are lagging in terms of the different characteristics of data. Let's now talk about big data solutions. Big data solutions are focused on combining all the data dimensions that were previously ignored or considered of minimum value, taking all the available sources and types into consideration and analyzing them for different and difficult-to-identify patterns.

Big data solutions are not just about the data itself or other characteristics of data; it is also about affordability, making it easier for organizations to store all of their data for analysis and in real time, if required. You may discover different insights and facts regarding your suppliers, customers, and other business competitors, or you may find the root cause of different issues and potential risks your organization might be faced with.

Big data comprises structured and unstructured datasets, which also eliminates the need for any other relational database management solutions, as they don't have the capability to store unstructured data or to analyze it.

Another aspect is that scaling up a server is also not a solution, no matter how powerful it might be; there will always be a hard limit for each resource type. These limits will undoubtedly move upward, but the rate of data increase will grow much faster. Most importantly, the cost of this high-end server and resources will be relatively high. Big data solutions comprise clustered computing mechanisms, which involve commodity hardware with no high-end servers or resources and can easily be scaled up and down. You can start with a few servers and can easily scale without any limits.

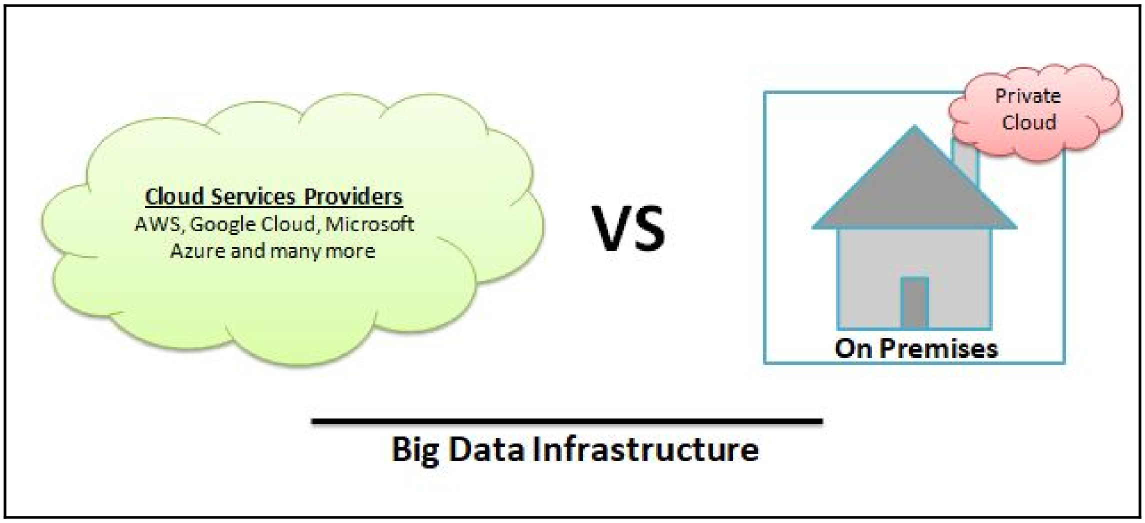
If we talk about just data itself, in big data solutions, data is replicated to multiple servers, commonly known as data nodes, based on the configurations done to make them fault tolerant. If any of the data nodes fail, the respective task will continue to run on the replica server where the copy of same data resides. This is handled by the big data solution without additional software development and operation. To keep the data intact, all the data copies need to be updated accordingly.

Distributed computing comprises commodity hardware, with reasonable storage and computation power, which is considered much less expensive compared to a dedicated processing server with powerful hardware. This led to extremely cost-effective solutions that enabled big data solutions to evolve, something that was not possible a couple of years ago.

Big data solutions – cloud versus on-premises infrastructure

Since the time when people started realizing the power of data, researchers have been working to utilize it to extract meaningful information and identify different patterns. With big data technology enhancements, more and more companies have started using big data and many are now on the verge of using big data solutions. There are different infrastructural ways to deploy a big data setup. Until some time ago, the only option for companies was to establish the setup on site. But now they have another option: a cloud setup. Many big companies, such as Microsoft, Google, and Amazon, are now providing a large amount of services based on company requirements. It can be based on server hardware configuration, and can be for computation power utilization or just storage space.

Later in this book, we will discuss these services in detail.



Different types of big data infrastructure

Every company's requirements are different; they have different approaches to different things. They do their analysis and feasibility before adopting any big changes, especially in the technology department. If your company is one of them and you are working on adopting any big data solution, make sure to bear the following in mind.

Cost

This is a very important factor, not just for small companies but also for big companies. Sometimes, it is the only deciding factor in finalizing any solution.

Setting up infrastructure on site has a big start up costs. It involves high-end servers and network setup to store and analyze information. Normally, it's a multimillion dollar setup, which is very difficult for small companies to bear; that would lead them to stay on a traditional approach. Now, with the emergence of technology, these companies have the option to opt for a cloud setup. In this way, they have the option to have an escape route from the start up costs, while utilizing the full potential of big data analytics. Companies offering cloud setup have very flexible plans, such as users having the option to just have a server and increase them as required. On the other hand, setup on site requires network engineers and experts to monitor full-time activity of the setup and maintain it. In a cloud setup, there is less hassle for companies; they just need to worry about how much storage capacity they need and how much computational power is required for analysis purposes.

Security

This is one of the main concerns for companies, because their establishment depends on it. Infrastructure setup on premises gives companies a sense of more security. It also gives them control over who is accessing their data, when it is used, and for what purpose it is being used. They can do their due diligence to make sure the data is secure.

On the other hand, data in the cloud has its inherent risks. Many questions arise with respect to data security when you don't know the whereabouts of your data. How is it being managed? Which team members from the cloud infrastructure provider can access the data? What, if any, unauthorized access was made to copy that data? That being said, reputable cloud infrastructure providers are taking serious measures to make sure that every bit of information you put on the cloud is safe and secure. Many encrypting mechanisms are being deployed so that even if any unauthorized access is made, the data will be useless for that person. Secondly, additional copies of your data are being backed up on entirely different facilities to make sure that you don't lose your data. Measures such as these are making cloud infrastructure almost as safe and secure as on-site setup.

Current capabilities

Another important factor to consider while implementing big data solutions in terms of on-premises setup versus cloud setup is whether you currently have big data personnel to manage your implementation on site. Do you already have a team to support and oversee all aspects of big data? Is it with your budget, or can you afford to hire them? If you are starting from scratch, the staff required for on-site setup will be significant, from big data architects to network support engineers. It doesn't mean that you don't need a team if you opt for cloud setup. You will still need big data architects to implement your solution so that companies can focus on what's important, making sense of the information gathered and effecting implementation in order to improve the business.

Scalability

When you install infrastructures for big data on site, you must have done some analysis about how much data will be gathered, how much storage capacity is required to store it, and how much computation power is required for analysis purposes. Accordingly, you must decide on the hardware required in this setup. Now in future, if your data analysis requirement changes, you may start receiving data from more sources or need more computation power to perform analysis on it. In order to fulfil this requirement, you need more servers or for additional hardware to be installed in your on-site setup.

Another aspect will be this: let's suppose you did the analysis that you will gather this much data, but after implementation, you are not receiving that much, or before, you assumed that you would need this much computational power but now, after implementation, you realize that you don't. In both cases, you will end up with expensive server hardware of no use to manage. In a cloud setup, you will have the option to scale your hardware up and down, without worrying about the negative financial implications, and doing so is incredibly easy.

We will now move on to briefly discuss some of the key concepts and terminologies that you will encounter in your day-to-day life while working in the world of big data.

Big data glossary

The section includes some key definitions to sum up the core concepts of big data. These will help you to keep key definitions in mind as we move further into the book, to help you understand what we have learned so far, and what lies ahead of us.

Big data

Data that is massive in volume, with respect to the processing system, with a variety of structured and unstructured data containing different data patterns to be analyzed.

Batch processing

A process of analyzing large datasets, which is typically scheduled and executes in bulk when no other processes are running. This is typically ideal for non-time-sensitive work that operates on very large datasets. Once the process has finished executing the task, it will return results typically as an output file or as a database entry.

Cluster computing

Cluster computing is the practice of combining the resources of multiple commodity low-cost hardwares and managing their collective processing and storage capabilities to execute different tasks. It requires a software layer to handle communication between different individual nodes in order to effectively manage and coordinate the execution of assigned work.

Data warehouse

A large repository of structured data for analysis and reporting purposes. It is composed of data that is already cleaned, with a definite schema and well integrated with sources. It is normally referred to in the context of traditional systems, such as BI.

Data lake

Similar to a data warehouse, for storing large datasets, but it comprises unstructured data. This is a commonly used term in the context of big data solutions that store information such as blogs, posts, videos, photos, and more.

Data mining

Data mining is the process of trying to process a mass of data into a more understandable and visual medium. It is a broad term for the practice of trying to find hidden patterns in large datasets.

ETL

ETL stands for **extract, transform, and load**. This mainly refers to traditional systems such as BI, which take raw data and process it for analytical and reporting purposes. It is mainly associated with data warehouses, but characteristics of this process are also found in the ingestion pipelines of big data systems.

Hadoop

An open source software platform for processing very large datasets in a distributed environment with respect to storage and computational power, mainly built on low-cost commodity hardware. It is designed for easy scale up from a few to thousands of servers. It will help to process locally stored data in an overall parallel processing setup. It comprises different modules–**Hadoop Distributed File System** (**HDFS**), Hadoop MapReduce, and Hadoop **YARN** (**Yet Another Resource Negotiator**).

In-memory computing

A strategy that involves moving the working datasets entirely within a cluster's collective memory instead of reading it from hard disk, to reduce the processing time while omitting I/O bound operations. Intermediate calculations are not written to disk and are instead held in memory. This is the fundamental idea of projects such as Apache Spark. Because of this, it has huge advantages in speed over I/O bound systems such as MapReduce.

Machine learning

The study that involves designing a system that can learn without being explicitly programmed. It can have the ability to adjust and improve itself based on the data fed to it. It involves the implementation of predictive and statistical algorithms that can continually zero in on *correct* behavior and insights as more data flows through the system.

MapReduce

MapReduce is a framework for processing any task in a distributed environment. It works on a Master/Slave principle similar to HDFS. It involves splitting a problem set into different nodes available in a clustered computing environment and produces intermediate results. It then shuffles the results to align like sets, and then reduces them by producing a single value for each set.

NoSQL

NoSQL provides a mechanism for storage and retrieval of data that is not in tabular form, such as what we store in relational database systems. It is also called non-SQL or a non-relational database. It is well suited to big data as it is mostly used with unstructured data and can work in distributed environments.

Stream processing

The practice of processing individual data items as they move through a system. This will help with real-time analysis of the data as it is being fed to the system. It is useful for time-sensitive operations using high velocity metrics.

Summary

In this chapter, we have learned about big data, its characteristics, the six Vs of big data, and the different aspects of big data. We have also discussed clustered computing and how to approach it for big data solutions. We also compared big data infrastructure on site versus the on-cloud setup. At the end, we briefly described some commonly used terms that you maybe already have encountered and will encounter, whenever you deal with big data. In the next chapter, we will go through the step-by-step setup of creating our own Hadoop environment, from a single node server to a clustered environment.

2 Big Data Environment Setup

In this chapter, we will go through a step by step guide on how to set up an environment to run big data applications. We will go through the following step-by-step setup guides:

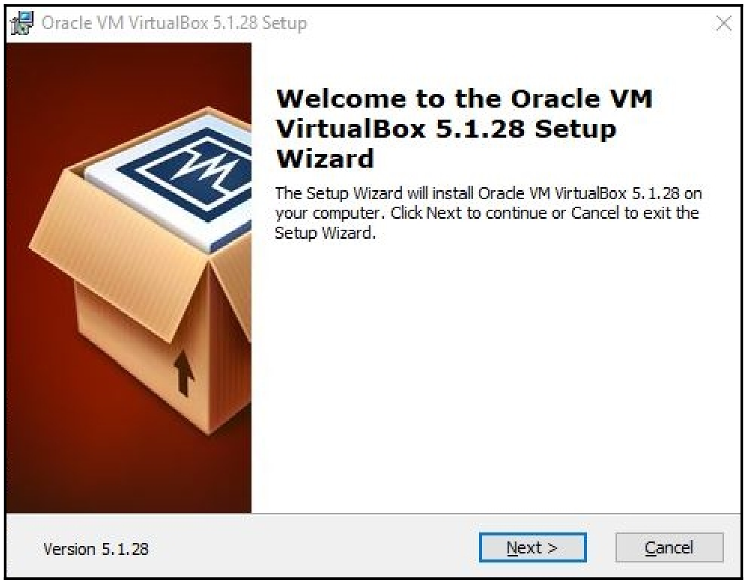
- Installing Oracle VM VirtualBox
- Installing and configuring Ubuntu
- Installing Apache Hadoop prerequisite applications
- Installing Apache Hadoop
- Configuring Hadoop to run as a single node server and pseudo-distributed mode

Oracle VM VirtualBox installation

There is a lot of software available in the market that can help you create virtual environments, also known as **virtual machines**, on top of your existing operating system. Creation of a virtual machine is useful if you want to try a new setup or application without affecting your existing configuration. Most of the softwares are available for all platforms, such as Windows, macOS, and different Linux distributions. Some of them are paid and some are free. We will be using Oracle VM VirtualBox, as it is a free open source software available under GNU **General Public License** (**GPL**) version 2:

1. Download the latest version of the application from `http://www.virtualbox.org/`, based on your operation system.
2. Double click on the downloaded file to start setup.

3. Click Next and leave the default options selected, as shown in the following screenshot:

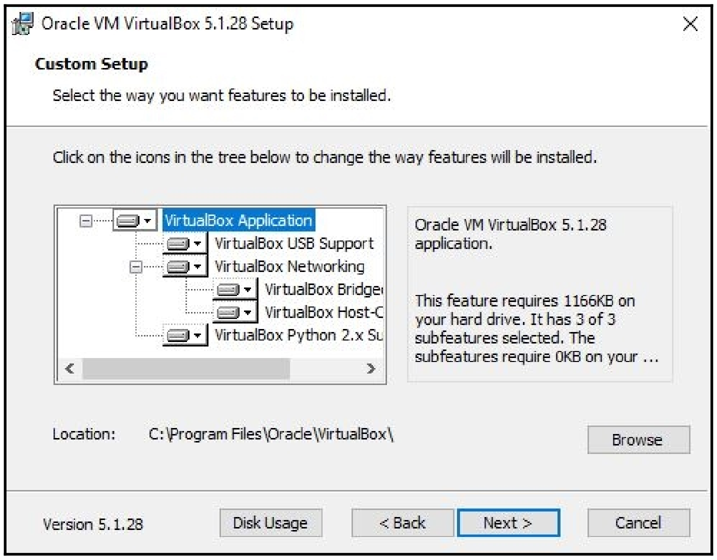

Oracle VM VirtualBox 5.1.28 Setup

Welcome to the Oracle VM VirtualBox 5.1.28 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 5.1.28 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 5.1.28

Figure 2.1.1


Oracle VM VirtualBox 5.1.28 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

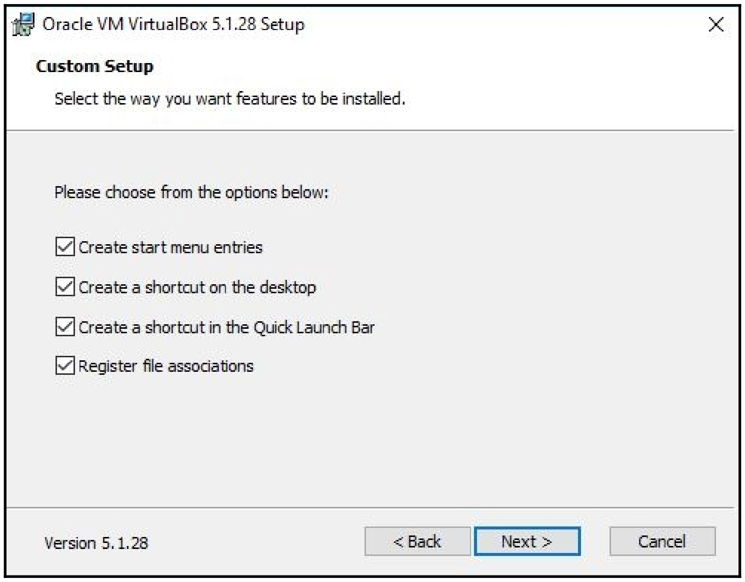
Oracle VM VirtualBox 5.1.28 application.

This feature requires 1166KB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 0KB on your ...

Location: C:\Program Files\Oracle\VirtualBox\

Version 5.1.28

Figure 2.1.2



Oracle VM VirtualBox 5.1.28 Setup

Custom Setup

Select the way you want features to be installed.

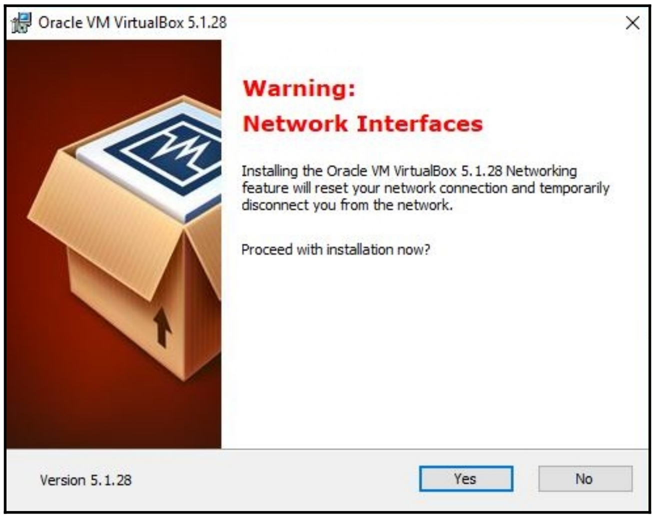
Please choose from the options below:

- [✓] Create start menu entries
- [✓] Create a shortcut on the desktop
- [✓] Create a shortcut in the Quick Launch Bar
- [✓] Register file associations

Version 5.1.28

Figure 2.1.3

4. It will show a warning sign that it will temporarily reset and disconnect your network in. Click **Yes** and continue with the setup, as shown in the following screenshot:



Oracle VM VirtualBox 5.1.28

Warning:
Network Interfaces

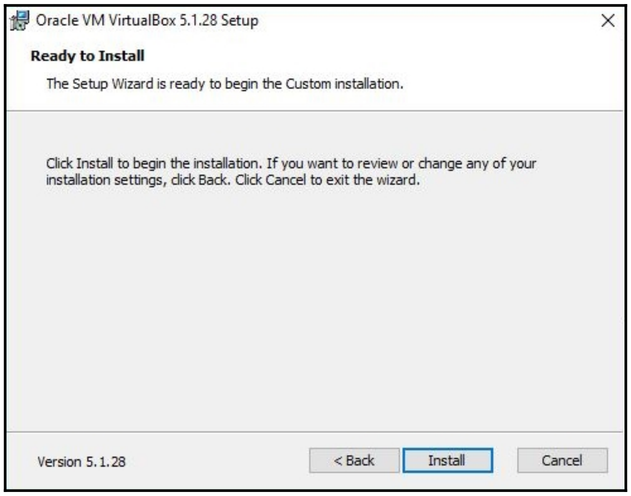
Installing the Oracle VM VirtualBox 5.1.28 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 5.1.28

Figure 2.1.4

5. Click **Next** until you finish the setup as follows:


Oracle VM VirtualBox 5.1.28 Setup

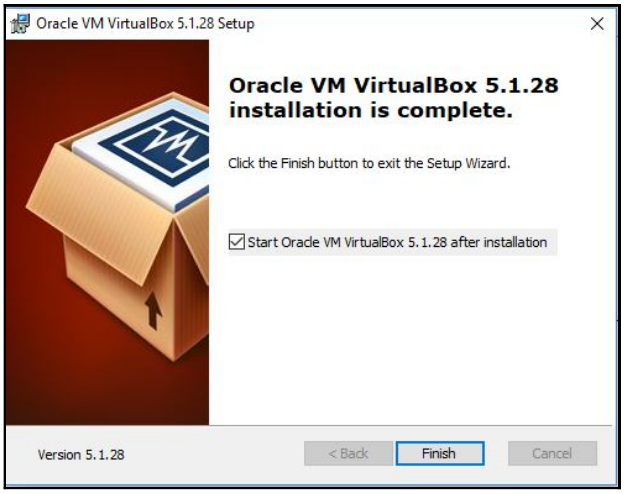
Ready to Install

The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Version 5.1.28

Figure 2.1.5


Oracle VM VirtualBox 5.1.28 Setup

Oracle VM VirtualBox 5.1.28 installation is complete.

Click the Finish button to exit the Setup Wizard.

☑ Start Oracle VM VirtualBox 5.1.28 after installation

Version 5.1.28

Figure 2.1.6

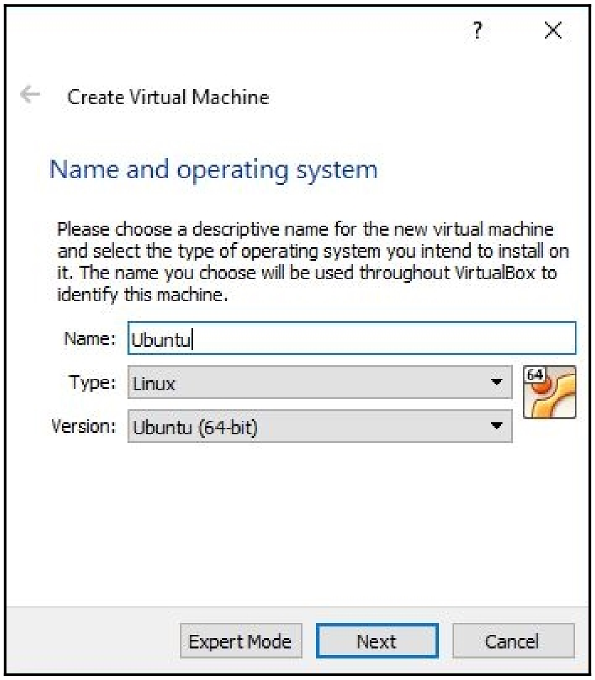
TIP

It is also recommended to install VirtualBox Extension pack, which gives you added features such as support for USB 2.0 and USB 3.0, disk encryption, on top of the default installation.

Ubuntu installation

After the successful installation of VirtualBox application, let's go ahead and create our first virtual machine and install Ubuntu Linux on it. Later, we will be using this Ubuntu Linux to configure big data applications and modules. Follow the steps given to set up the VM and install Ubuntu:

1. Download the Ubuntu ISO file from `https://www.ubuntu.com/download/desktop`.
2. Open the VirtualBox application and click on **New**. Give a **Name** to your new virtual machine as shown in the following screenshot:



Create Virtual Machine

Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

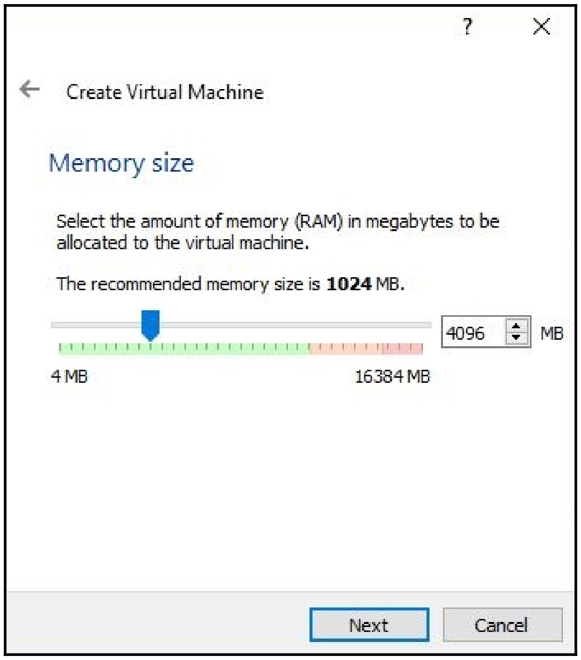
Name: Ubuntu

Type: Linux

Version: Ubuntu (64-bit)

Figure 2.2.1

3. Assign memory (RAM). It is recommended to assign 4GB, as shown in the screenshot that follows:


Create Virtual Machine

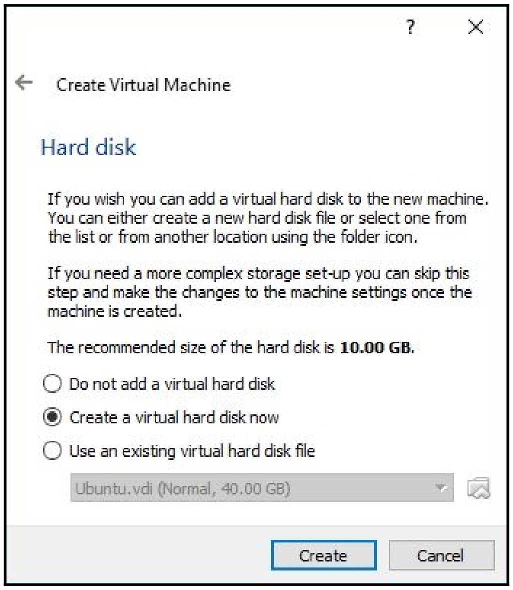
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

Figure 2.2.2

4. The next step is to **Create a virtual hard disk now** for your VM, as shown in the screenshot that follows:



Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

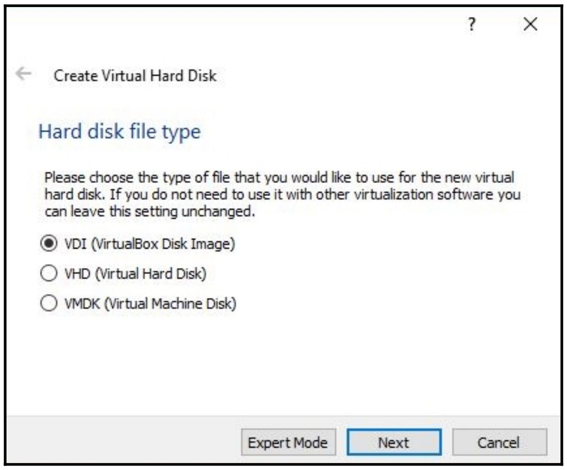
The recommended size of the hard disk is **10.00 GB**.

- ○ Do not add a virtual hard disk
- ● Create a virtual hard disk now
- ○ Use an existing virtual hard disk file

Ubuntu.vdi (Normal, 40.00 GB)

Figure 2.2.3

5. It will ask you to select the type of virtual hard disk you want to create; keep the default option selected (**VDI** (**VirtualBox Disk Image**)) and click **Next** to continue, as shown in the following screenshot:



Create Virtual Hard Disk

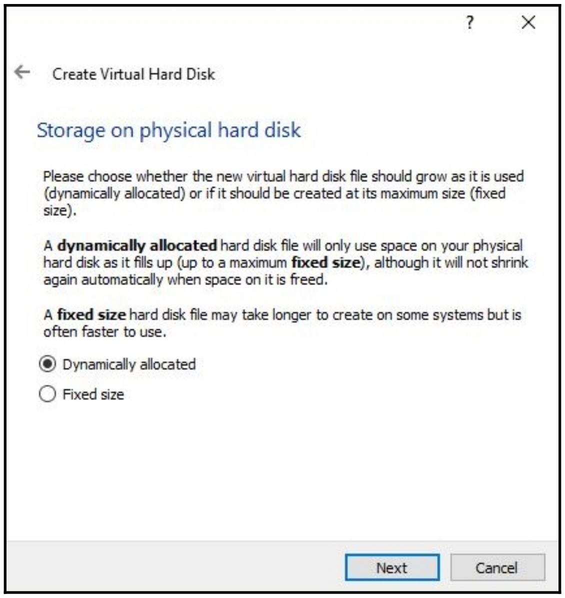
Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ● VDI (VirtualBox Disk Image)
- ○ VHD (Virtual Hard Disk)
- ○ VMDK (Virtual Machine Disk)

Figure 2.2.4

6. It will give you an option as to whether the hard disk will be dynamically allocated or of a fixed size. If you select fixed size and assign some limits to it, you will no longer be able to increase the hard disk size in future. It's better to select the dynamically allocated option, as shown in the screenshot. Click **Next** to continue:



Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

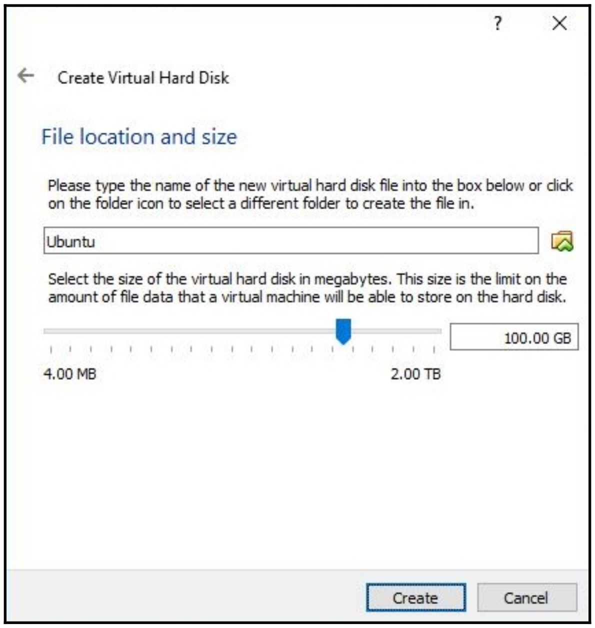
A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

[✓ Dynamically allocated]
Fixed size

Figure 2.2.5

7. It will ask you to assign an initial hard disk size. Later, VirtualBox will increase it automatically as the initial assigned space gets utilized. Click **Create**:


Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

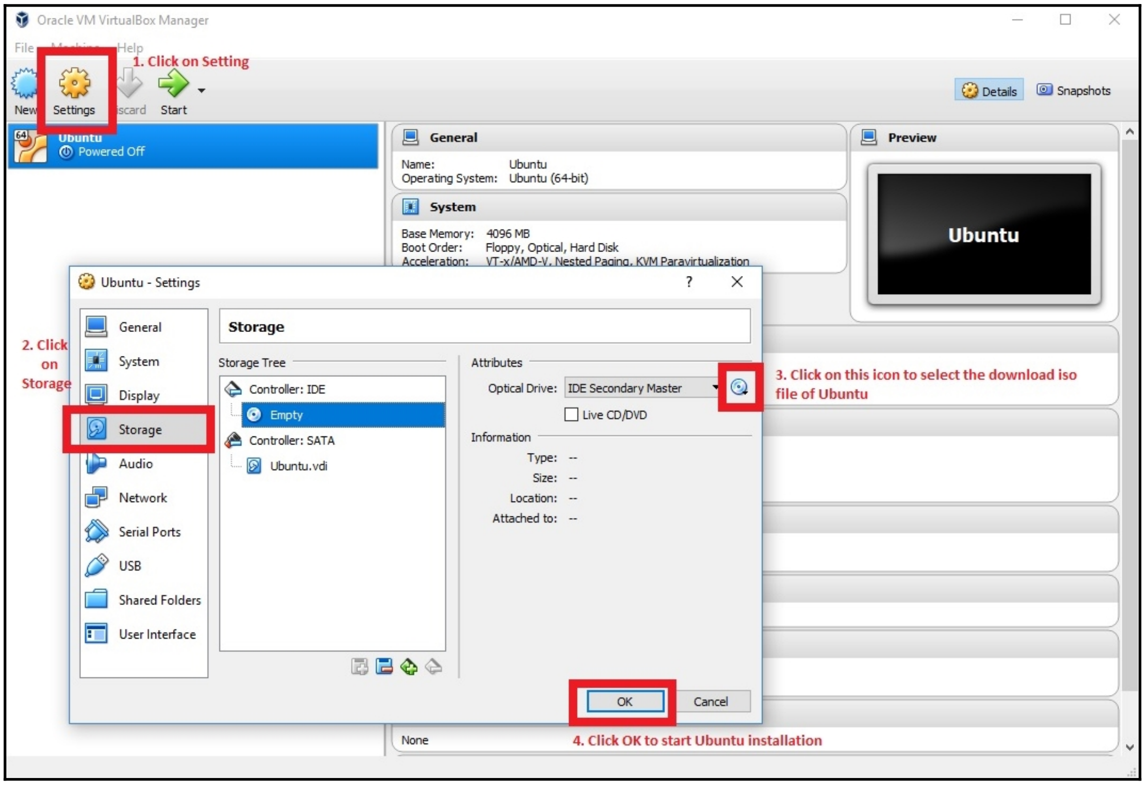
Ubuntu

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

100.00 GB

Figure 2.2.6

8. Your new virtual machine is now created. It's time to assign the bootable ISO file that we downloaded in the first step, to make the virtual machine bootable. It's a four step process, as the following screenshot shows:



Name: Ubuntu
Operating System: Ubuntu (64-bit)

Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Storage

Storage Tree
Controller: IDE
Empty
Controller: SATA
Ubuntu.vdi

Attributes
Optical Drive: IDE Secondary Master
Live CD/DVD

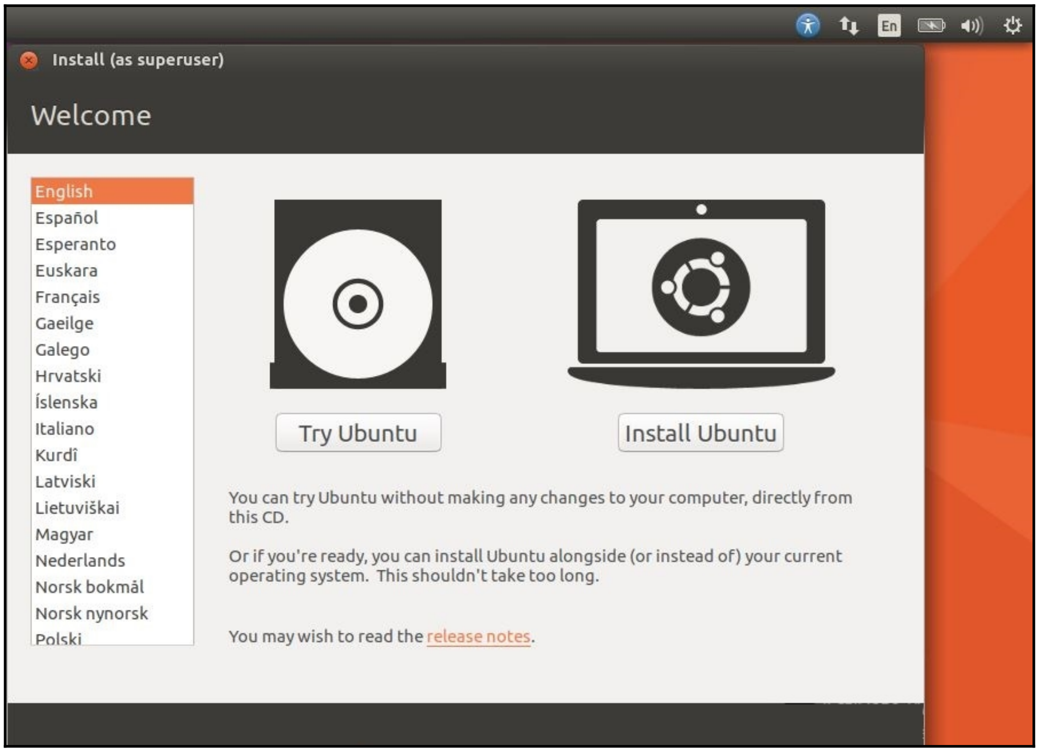
Information
Type: --
Size: --
Location: --
Attached to: --

None

Figure 2.2.7

1. Select your Virtual Machine, **Ubuntu**, and click on the Settings icon.
2. Select the Storage menu item from the left menu list.
3. Click on the CD-like icon to select the Ubuntu ISO file that was downloaded in *Step 1*.
4. Click OK to close the Settings dialogue box.

All steps are completed in terms of setting up the new Virtual Machine. Now it's time to start the Ubuntu installation. Select **Ubuntu** and click on the **Start** button from the top menu bar to launch your virtual machine. At the start, you will get a few prompts regarding keyboard and mouse control; keep pressing the **OK** button until you see the Ubuntu setup screen, as shown in the screenshot that follows:



You can try Ubuntu without making any changes to your computer, directly from this CD.

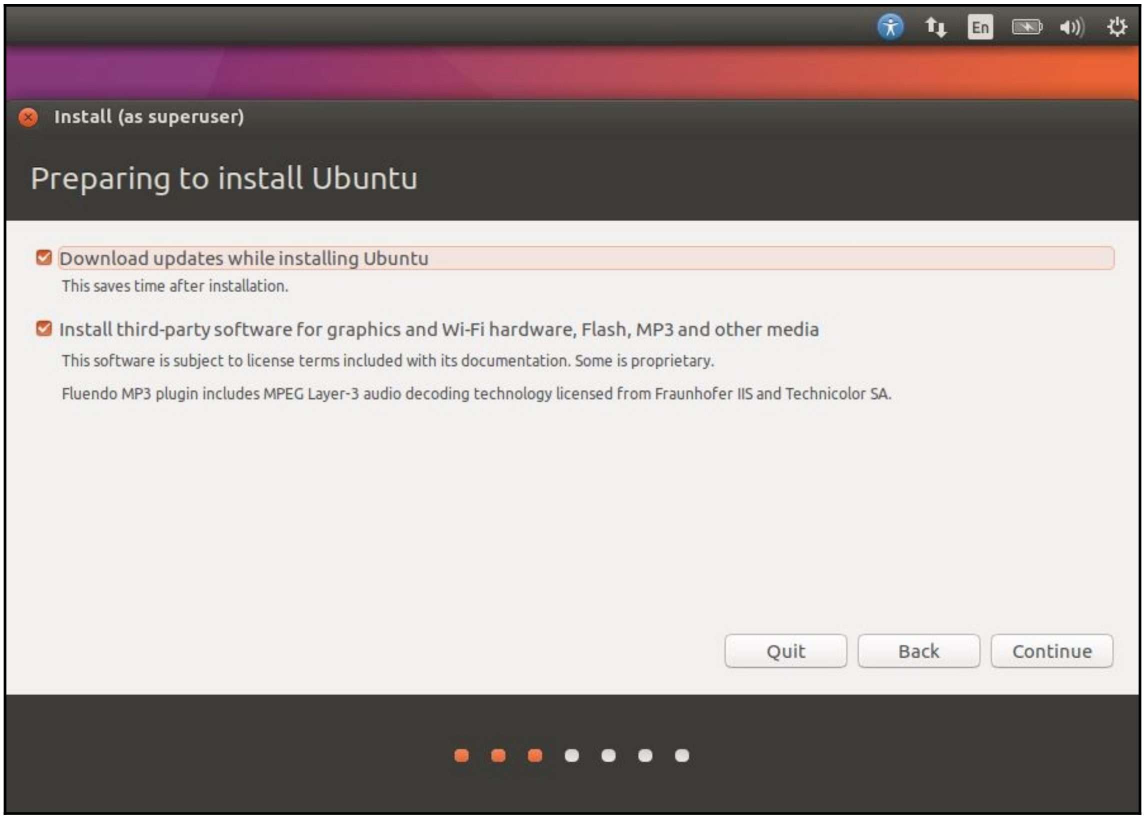
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

Figure-2.3.1

We will now go through the following step by step guide to install Ubuntu Linux:

1. Click on **Install Ubuntu** to proceed to next step. The setup will give the option to automatically download the latest updates and some third party drivers and applications with respect to the current Ubuntu version during the installation process, as shown in the screenshot that follows:


Install (as superuser)

Preparing to install Ubuntu

☑ Download updates while installing Ubuntu

This saves time after installation.

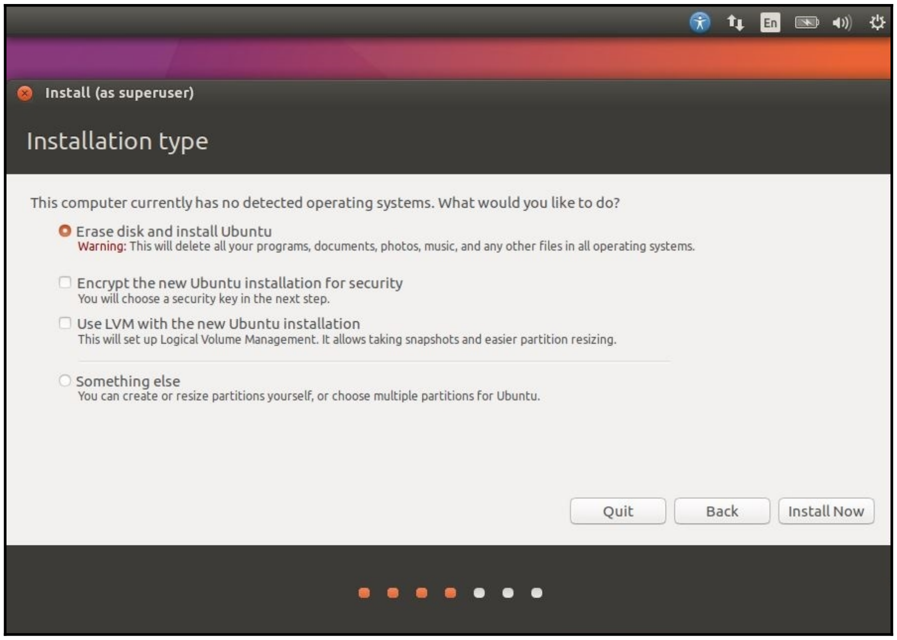
☑ Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media

This software is subject to license terms included with its documentation. Some is proprietary.

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Figure-2.3.2

2. This step gives an option as to how you want to partition your hard disk. We are currently installing it in a virtual environment and no other operating system is already installed, so let the setup itself partition your hard drive with the standard options, as shown in the image that follows. It also gives you an option to encrypt the data that resides in your hard disk. To enable this feature, select the **Encrypt the new Ubuntu installation for security** option. We will not be selecting this option for now, so proceed further. The setup will show a confirmation dialog that the changes will be written to the disk.
   Click **Continue** to proceed, as seen in the screenshot that follow:



Installation type

This computer currently has no detected operating systems. What would you like to do?

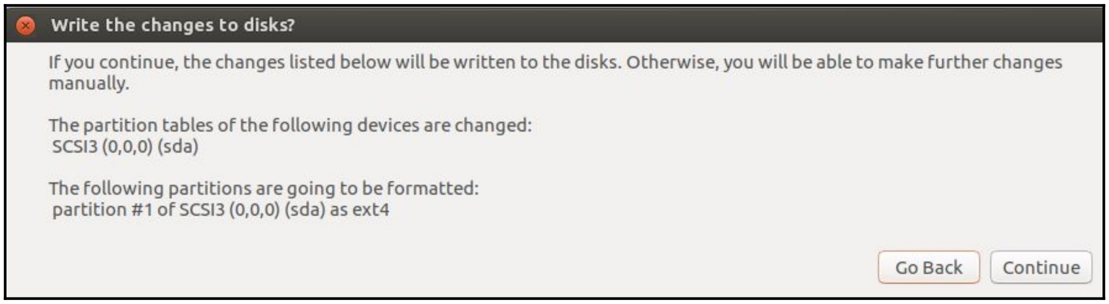
[✓ Erase disk and install Ubuntu]
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

☐ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

☐ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Figure-2.3.3



Write the changes to disks?

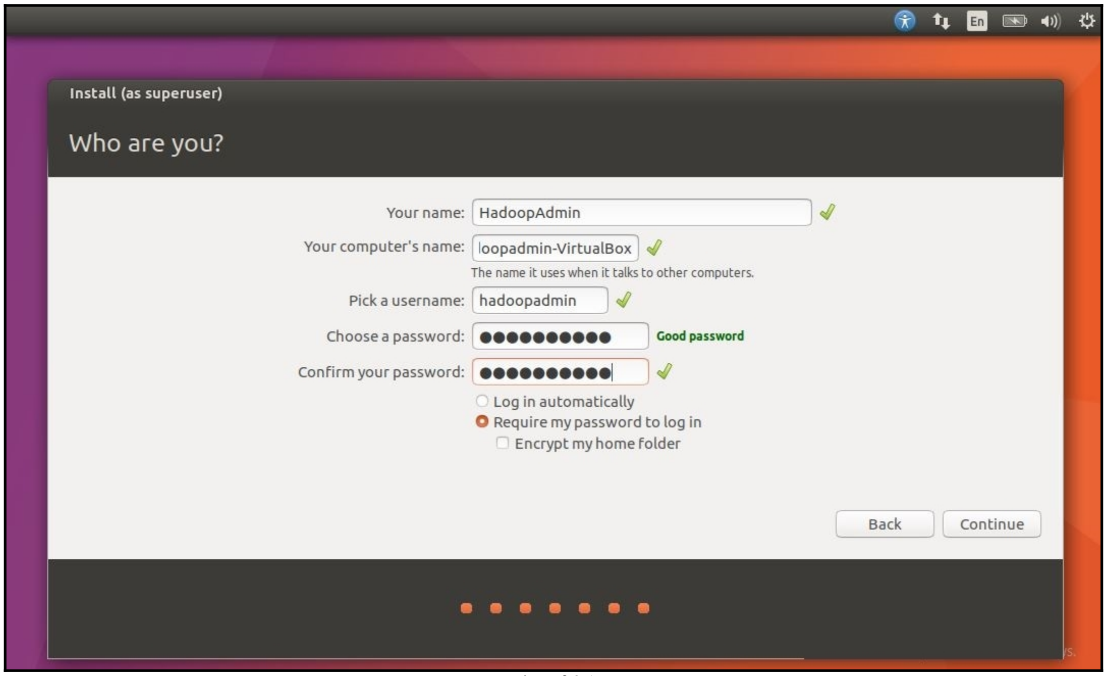
If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.

The partition tables of the following devices are changed:
SCSI3 (0,0,0) (sda)

The following partitions are going to be formatted:
partition #1 of SCSI3 (0,0,0) (sda) as ext4

Figure-2.3.4

3. The next step is to select your timezone. Fill out your location and click **Continue**.
4. The next step is to select your keyboard layout. If it is anything other than English, select the required language and keyboard layout settings. Otherwise, click **Continue** to proceed to the next step.
5. This step covers configuring your machine; give your computer a name and create your username and password. We will be using username **HadoopAdmin,** as shown in the following screenshot:



Install (as superuser)

Who are you?

Your name: HadoopAdmin ✓
Your computer's name: loopadmin-VirtualBox ✓
The name it uses when it talks to other computers.
Pick a username: hadoopadmin ✓
Choose a password: ●●●●●●●●●● Good password
Confirm your password: ●●●●●●●●●● ✓
Log in automatically
[✓ Require my password to log in]
Encrypt my home folder

Figure-2.3.5

6. Click **Continue** to start the installation process. Once the installation finishes, the setup will ask you to restart your computer. After restarting, you will see a screen similar to the following figure. This means that you have successfully installed Ubuntu on your virtual machine. It will look something like this:

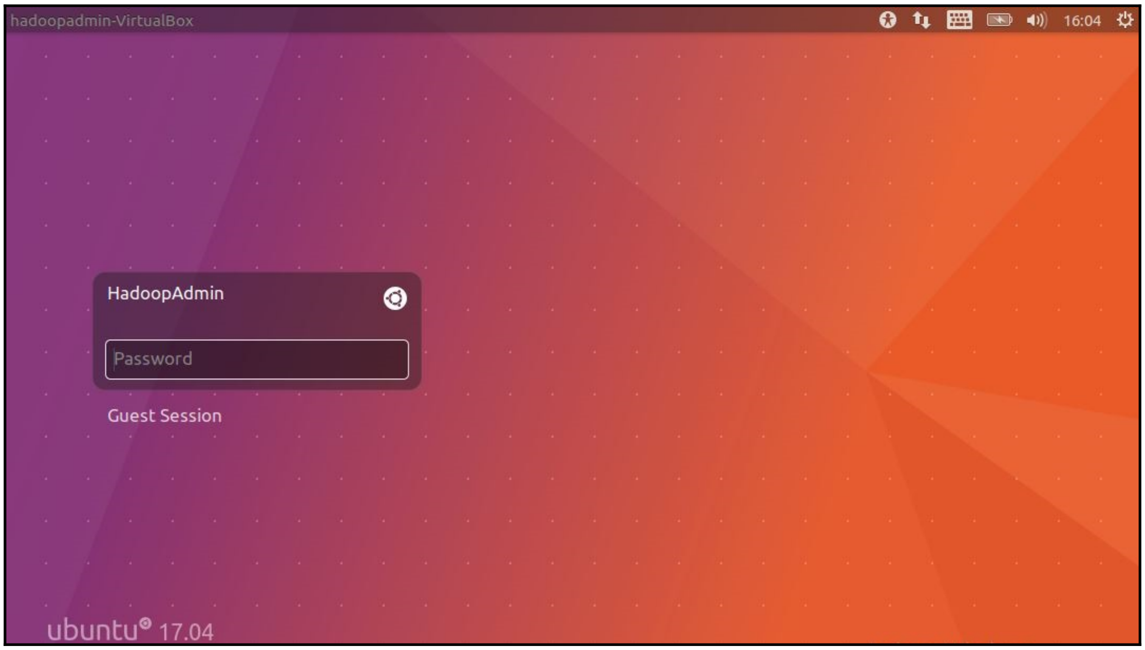


Figure-2.3.6

Hadoop prerequisite installation

Hadoop modules are mostly written in Java, so one of the prerequisites is to install Java. Hadoop daemons requires SSH to encrypt the communication between its nodes in an insecure environment. We will now go through a step by step guide for installing and configuring Java and SSH.

Java installation

Hadoop is compatible with both Oracle Java and OpenJDK, but it is preferable to install Oracle Java. To install it, use the following instructions:

1. Download JDK and JRE from `http://www.oracle.com/technetwork/java/javase/downloads/index.html`.
2. Copy the downloaded `tar.gz` file to the location where you want to install Java.
3. Extract and install Java by using the following command. Make sure to replace `8u144` with the Java version you have downloaded:

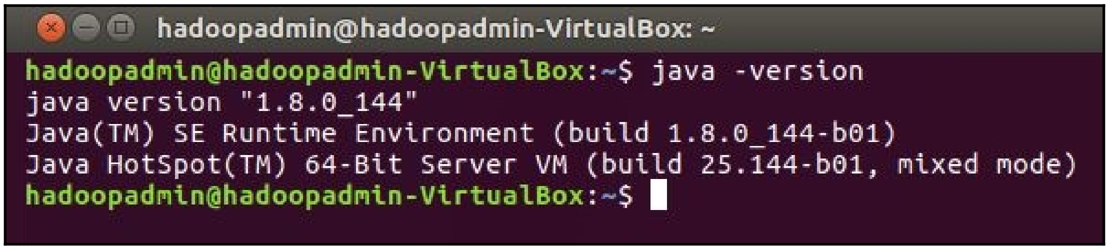
   ```
   $ tar zxvf jdk-8u144-linux-x64.tar.gz
   ```

4. The next step is to point the system to the extracted directory. In our case, files are extracted to the `;/home/hadoopadmin/jdk1.8.0_144/` folder. Execute the following command to make this Java version the default JVM:

   ```
   $ sudo update-alternatives --install /usr/bin/java java
   /home/hadoopadmin/jdk1.8.0_144/bin/java 100
   $ sudo update-alternatives --install /usr/bin/javac javac
   /home/hadoopadmin/jdk1.8.0_144/bin/javac 100
   ```

5. To make sure Java is installed correctly, use the following command to check your Java version. It will display an output similar to the image that follows the code block:

   ```
   $ java -version
   ```

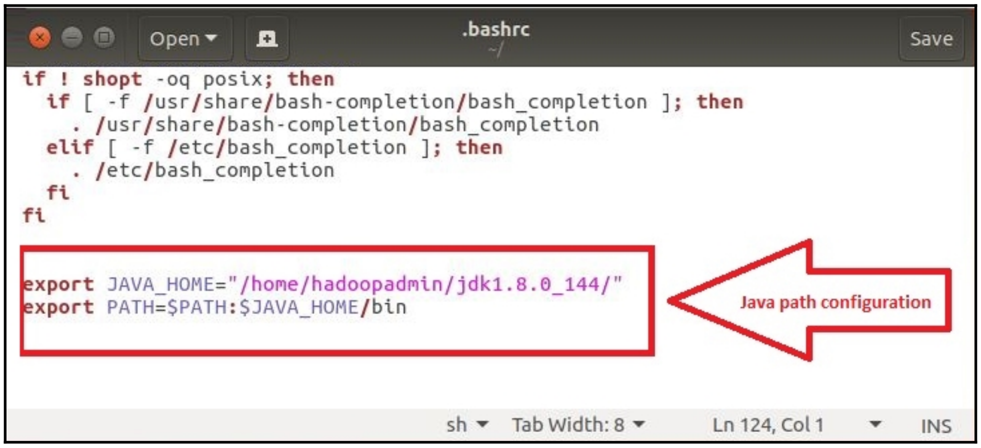


```
hadoopadmin@hadoopadmin-VirtualBox:~$ java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Figure-2.4.1

6. Edit the `.bashrc` file using the following command, and configure the Java path at the end of the file. Make sure not to make any other changes in this file. The image that follows the code block will give you a better idea:

   ```
   $ sudo gedit ~/.bashrc
   ```



```
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export JAVA_HOME="/home/hadoopadmin/jdk1.8.0_144/"
export PATH=$PATH:$JAVA_HOME/bin
```

Figure-2.4.1.1

7. Once the installation finishes, you can delete the downloaded file as it is no longer required.

Regular updates are expected of any application. Similarly, Java versions tend to update. When installing Java for Hadoop, make sure to refer to the wiki site of Hadoop to check the best matching Java version for your Hadoop installation. The wiki site is available at `https://wiki.apache.org/hadoop/HadoopJavaVersions`.

SSH installation and configuration

One of the prerequisites to run Hadoop is to install `ssh`. Hadoop scripts requires `sshd` to be running to manage Hadoop remote daemons. To install it in Ubuntu, use the following command:

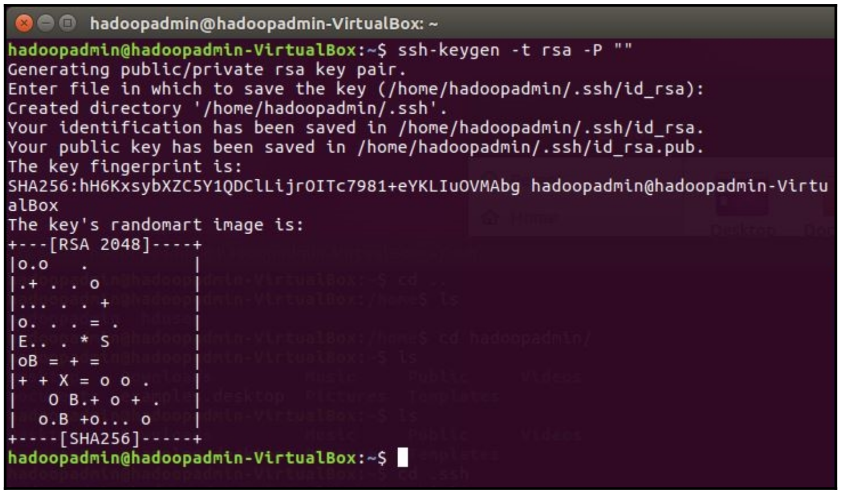
```
$ sudo apt-get install ssh
$ sudo apt-get install rsync
```

SSH, also known as **Secure Socket Shell** or **Secure Shell**, is a cryptographic protocol that helps to encrypt communication in unsecured networks where an SSHD is the daemon program for SSH. Together they provide secure communication between two untrusted hosts over an insecure network.

Once the installation completes after executing the preceding commands, it's time to configure your SSH services. The first step is to generate an SSH key for our user by executing the following command:

```
$ ssh-keygen -t rsa -P ""
```

`-t` is the type of key and `-P` implies for the password. It is left blank so that Hadoop can interact with its nodes with any user involvement. Otherwise every time Hadoop requires any communication with its nodes, the user has to enter a passphrase to continue the operation. The output of the preceding program is shown in the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoopadmin/.ssh/id_rsa):
Created directory '/home/hadoopadmin/.ssh'.
Your identification has been saved in /home/hadoopadmin/.ssh/id_rsa.
Your public key has been saved in /home/hadoopadmin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:hH6KxsybXZC5Y1QDClLijrOITc7981+eYKLIuOVMAbg hadoopadmin@hadoopadmin-Virtu
alBox
The key's randomart image is:
+---[RSA 2048]----+
|o.o    .         |
|.+ . . o         |
|... . . +        |
|o. . . = .       |
|E.. . * S        |
|oB = + =         |
|+ + X = o o .    |
|   O B.+ o + .   |
|  o.B +o... o    |
+----[SHA256]-----+
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Figure-2.4.2

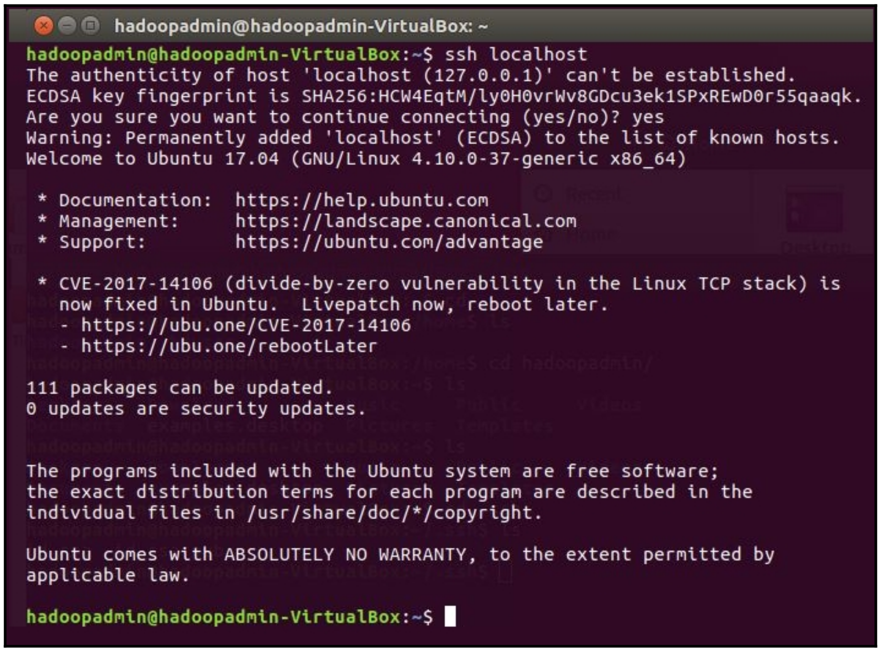
The next step is to enable SSH access locally using the previously generated keys. For this purpose, execute the following command:

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

It's time to test our configuration. User interaction is required only for the first time, to accept the connection and add a host key fingerprint to the user's `known_hosts` file. Execute the following command to connect to your local machine using SSH without any user involvement:

```
$ ssh localhost
```

The output of the command is as follows:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:HCW4EqtM/ly0H0vrWv8GDcu3ek1SPxREwD0r55qaaqk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 17.04 (GNU/Linux 4.10.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * CVE-2017-14106 (divide-by-zero vulnerability in the Linux TCP stack) is
   now fixed in Ubuntu.  Livepatch now, reboot later.
   - https://ubu.one/CVE-2017-14106
   - https://ubu.one/rebootLater

111 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hadoopadmin@hadoopadmin-VirtualBox:~$
```

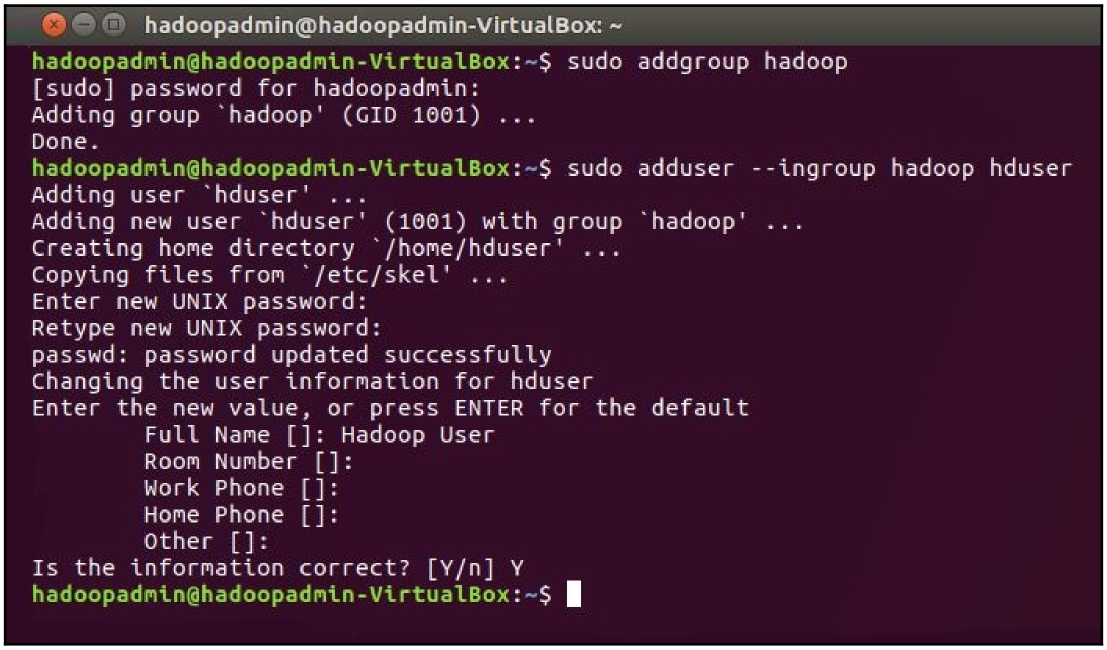
Figure-2.4.3

Hadoop system user

Adding a Hadoop system user is not required, but it is recommended. It helps to assign different sets of privileges and the permission to separate the Hadoop installation from other applications running on the same server. Execute the following commands to add a group and a user:

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

The system will ask you to create a password for the new user and some other information, such as `Full Name`, `Phone Number`, and so on. The output of the preceding command will be something similar to the screenshot that follows:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo addgroup hadoop
[sudo] password for hadoopadmin:
Adding group `hadoop' (GID 1001) ...
Done.
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
        Full Name []: Hadoop User
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Figure-2.4.4

In our case, this is an optional step, as we are using a dedicated virtual machine for the purpose of running only Hadoop. So, adding a separate privileged group and user is not required.

Apache Hadoop installation

After the successful installation of the prerequisites, it is now time to start the installation and configuration of Hadoop itself. The following are the steps to get started:

1. Download the stable version of Apache Hadoop from `http://www-us.apache.org/dist/hadoop/common/`. As of now the stable version available for download is `hadoop-2.8.1`.

2. Extract the downloaded file by using the following command:

   ```
   $ tar zxvf hadoop-2.8.1.tar.gz
   ```

3. It's time to configure some parameters to run Hadoop. Use the following command to edit the `hadoop-env.sh` configuration file:

   ```
   $ gedit etc/hadoop/hadoop-env.sh
   ```

4. Look for the following line to set the `JAVA_HOME` path. Replace `/home/hadoopadmin/jdk1.8.0_144/` with the directory where you have installed it. In our case, it will remain the same:

   ```
   # set to the root of your Java installation
   export JAVA_HOME=/home/hadoopadmin/jdk1.8.0_144/
   ```

5. It's time to test the basic configuration of Hadoop. Try to run the following command in your terminal:

   ```
    $ bin/hadoop
   ```

If you get some usage documentation of the Hadoop script, it means it is configured correctly. If not, it may be that you have made a mistake in one of the preceding steps. Try to go through them again to make sure every step has been followed correctly.

Hadoop configuration

Configuration files come with the Hadoop package and are configured by default to run as a Single Node, as a single Java Process. There is also an option to run each Hadoop daemon as a separate Java process, which is also known as a pseudo-distributed operation. To set up pseudo-distributed mode, the following configurations are required. You can use any text editor of your choice; we will be using gedit. All configuration files are located in `etc/hadoop/` inside the extracted Hadoop package folder. In our case, the complete path is `/home/hadoopadmin/hadoop-2.8.1/etc/hadoop/`:

1. Edit the `core-site.xml` configuration file and copy the following code in between the `<configuration>..</configuration>` tabs:

   ```
   <property>
           <name>fs.defaultFS</name>
           <value>hdfs://localhost:9000</value>
   </property>
   ```

2. Edit the `hdfs-site.xml` file and copy the following in between the `<configuration>..</configuration>` tabs. You can change the value from `1` to any number; it depends how many copies of the data you want to replicate:

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
```

Path configuration for Hadoop commands

We will now configure the path for the Hadoop bin directory. As a result of this, you will be able to execute Hadoop commands globally from any path. This configuration is similar to the one which we did for the Java installation, in step 6. To do this configuration, execute the following command:

```
$ sudo gedit ~/.bashrc
```

Once you execute this command, it will open up a configuration file. Add the following code at the end of the file:

```
# Hadoop Environment
export HADOOP_HOME="/home/hadoopadmin/hadoop-2.8.1/"
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

Hadoop server start and stop

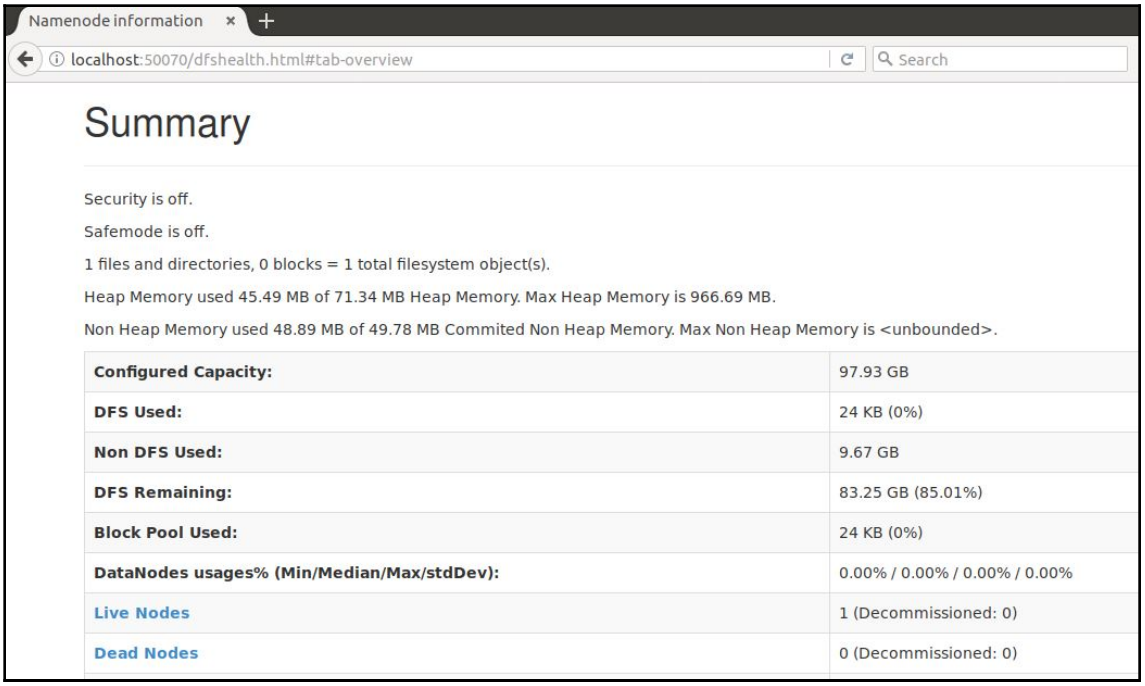
Now it's time to start our Hadoop server. But, before we proceed further, it's a part of the recommended practices to format NameNode for the first time before we do so. NameNode contains the metadata of the directory tree of all the files in DataNodes. The purpose of formatting NameNode is to utilize all the space available in DataNodes even if any unutilized data is stored. Execute the following command to format it:

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

To start your Hadoop server, execute the following command:

```
$ sbin/start-dfs.sh
$ sbin/start-yarn.sh
```

Open your browser and type in `http://localhost:50070/` to see an overview of your Hadoop Server. This web interface gives a lot of useful information, including a NameNode overview, the number of live DataNodes, the number of dead DataNodes, your overall storage capacity and its utilization, startup progress, and access to log files. The web interface will be similar to the following figure:



Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 45.49 MB of 71.34 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 48.89 MB of 49.78 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

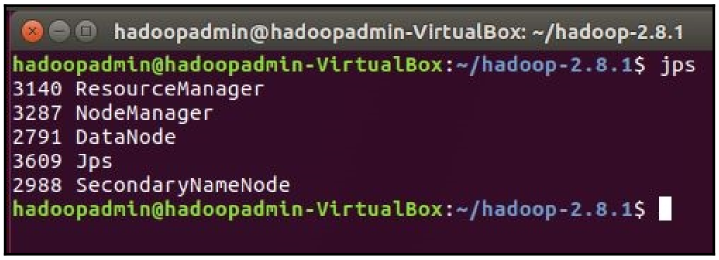
Configured Capacity:	97.93 GB
DFS Used:	24 KB (0%)
Non DFS Used:	9.67 GB
DFS Remaining:	83.25 GB (85.01%)
Block Pool Used:	24 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)

Figure-2.4.5

For Linux terminals, you can also verify that all services are running by executing the following command:

```
$ jps
```

You will get an output similar to the following image:



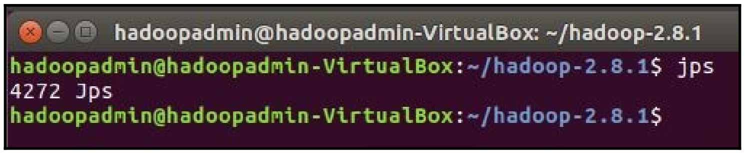
```
hadoopadmin@hadoopadmin-VirtualBox: ~/hadoop-2.8.1
hadoopadmin@hadoopadmin-VirtualBox:~/hadoop-2.8.1$ jps
3140 ResourceManager
3287 NodeManager
2791 DataNode
3609 Jps
2988 SecondaryNameNode
hadoopadmin@hadoopadmin-VirtualBox:~/hadoop-2.8.1$
```

To stop the Hadoop server, execute the following command:

```
$ sbin/stop-dfs.sh
$ sbin/stop-yarn.sh
```

You can verify whether all services related to Hadoop have been stopped by again running the jps command. You will get an output similar to Figure-2.4.7. Now, you will not see any of the services, such as DataNode, ResourceManager, and so on in the list:

```
$ jps
```



```
hadoopadmin@hadoopadmin-VirtualBox: ~/hadoop-2.8.1
hadoopadmin@hadoopadmin-VirtualBox:~/hadoop-2.8.1$ jps
4272 Jps
hadoopadmin@hadoopadmin-VirtualBox:~/hadoop-2.8.1$
```

Figure-2.4.7

Summary

In this chapter, we have gone through a step by step guide for installing and setting up a new virtual machine. Then, while using our new virtual machine, we have gone through the installation of Ubuntu to treat it as a dedicated server. We then learnt how to install and configure Apache Hadoop in a single node environment as well as in pseudo-distributed mode. In between, we have also gone through a step by step guide to the installation of the Hadoop prerequisite applications Java and SSH. Now, we are ready to go deeper into the big data world, and will be able to execute our programs.

In the next chapter, we will take a deep dive into the Hadoop echo system, considering the past and present of Hadoop, HDFS, MapReduce, and YARN in detail.

3
Hadoop Ecosystem

We should now have an understanding of what big data is and how it will impact industries in their decision-making. We have also created and configured our own big data virtual environment so that we can move forward in practical terms and build our own applications. These applications will get more and more mature as we proceed further in this book.

In this chapter, we will look at the Apache Hadoop Project in detail, touching on why it was developed and what the advantages and disadvantages of using Hadoop are. We will also go through the different components and modules of the Hadoop system and will also get our hands dirty with some practical activities. The main topics that we are going to cover in this chapter are as follows:

- Apache Hadoop
- HDFS
- MapReduce Framework
- YARN

We will then take a look at other Apache Projects that are directly or indirectly related to the world of big data world that fit into the Hadoop Ecosystem. Some of the projects that we will discuss include the following:

- Zookeeper
- Kafka
- Flume
- Cassandra
- HBase
- Spark

In this chapter, we will briefly discuss these projects' purposes and how they will impact your decision-making while designing your big data solution.

Apache Hadoop

Apache Hadoop is an open source software platform that processes very large datasets in a distributed environment with respect to storage and computational power, and is mainly built on low cost commodity hardware. It came into existence thanks to a Google File System paper that was published in October 2003. Another research paper from Google MapReduce looked at simplified data processing in large clusters.

Apache Nutch is a highly-scalable and extensible open source web crawler project which implemented the MapReduce facility and the distributed file system based on Google's research paper. These facilities were later announced as a sub-project called **Apache Hadoop**.

Apache Hadoop is designed to easily scale up from a few to thousands of servers. It helps you to process locally stored data in an overall parallel processing setup. One of the benefits of Hadoop is that it handles failure at a software level. The following figure illustrates the overall architecture of the Hadoop Ecosystem and where the different frameworks are within it:

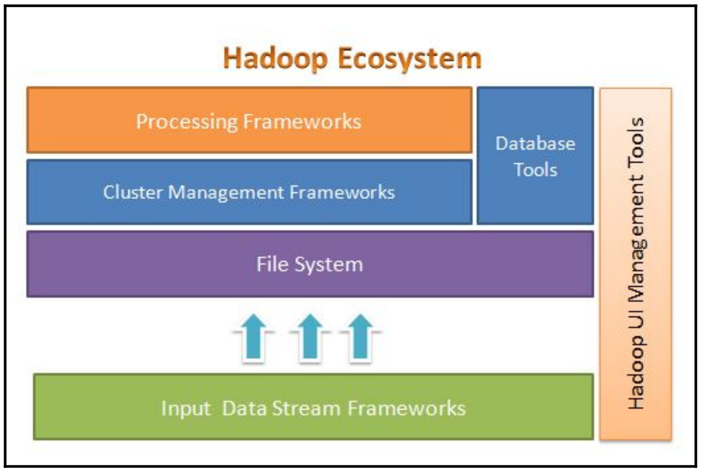

Hadoop Ecosystem
Processing Frameworks
Database Tools
Cluster Management Frameworks
File System
Input Data Stream Frameworks
Hadoop UI Management Tools

Figure-3.0.1

In Hadoop Ecosystem, Apache Hadoop provides a framework for the file system layer, cluster management layer, and the processing layer. It leaves an option for other projects and frameworks to come and work alongside Hadoop Ecosystem and develop their own framework for any of the layers available in the system. We will discuss other Apache Projects that fit in the remaining layers later.

Apache Hadoop is comprised of four main modules. These modules are Hadoop Distributed File System (the file system layer), Hadoop MapReduce (which works with both cluster management and the processing layer), **Yet Another Resource Negotiator** (**YARN**, the cluster management layer), and Hadoop Common. We will discuss these in turn as we proceed with this chapter.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a file system, as the name states, similar to Windows (FAT or NTFS) and Linux (swap, ext3, and so on) that is specifically designed for distributed systems that need to process large datasets with high availability. It is optimized to work with multiple nodes and spread data across them in a clustered computing environment to assure the maximum availability of data. The main difference between HDFS and other distributed file systems is that it is designed to run in a low-cost hardware setup and is highly fault-tolerant. The basic architecture of HDFS is shown in the following diagram:

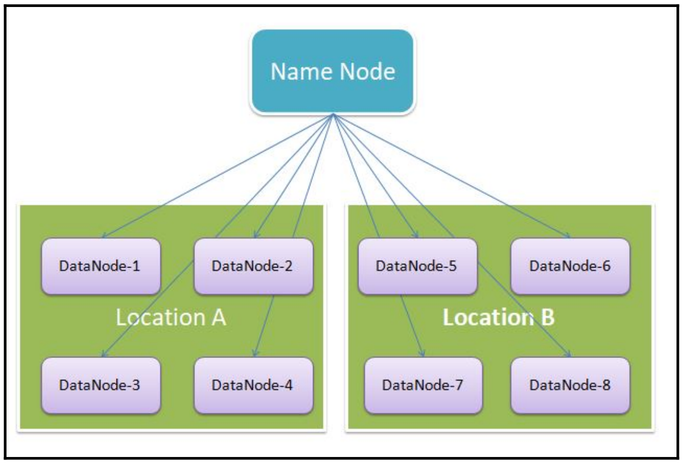


Figure-3.1.1

HDFS has two main components that basically work using the master and slave concept. These two components are called the **Name Node**, which is the master node, and the **DataNode**, which is a slave node.

NameNode is the key for the whole file system. It is responsible for spreading the data across different data nodes based on the defined configuration. It contains and manages metadata of the data blocks available across different data nodes. You might be thinking, 'What is metadata?' Metadata is the information related to files stored in a system, for example, the name of a file, the size of a file, into how many block a file was split, where its blocks are stored in the DataNodes, and so on.

Now consider what might happen if we lose this metadata information. Will we still be able to retrieve all of the information and files that are stored in the DataNodes? The answer is no. This is because we don't know which data block belongs to which file, and we even don't know how many data blocks there are for a single file. Without this metadata information, the whole HDFS system will collapse; that is why NameNode is considered to be a single point of failure.

To overcome this issue, it is recommended that we have reliable server hardware to act as a NameNode. If we rely on commodity hardware for NameNode, the chances of failure will increase. We use commodity hardware in DataNode because we are replicating data in three or more data nodes, so if any data node fails, we will still be able to extract the information from another data node. In the case of NameNode, however, the information is not being replicated anywhere; if it fails, the whole system will fail.

There are some configuration points that we have to take care of while setting up HDFS, such as how many times data has to be replicated, determining the block size of the data, and so on. The default block size in HDFS is 64 MB. This can be set to different sizes such as 128 MB, 256 MB, and so on. Normally, in other file systems, the data block size is relatively low -4 KB or 8 KB - but HDFS is designed to handle large datasets. If we reduce the size of the data block to a few KBs or MBs, the metadata information will increase, which in turn increases the load on NameNode while retrieving file information. It will also increase the overhead. In other words, when we need to execute a job or a program, the more data, network activity, and processing will be required.

DataNode is responsible for a fragment of a large dataset commonly known as a data block. Local storage management is assigned by NameNode. The default configuration of HDFS copies each data block into three different data nodes, one on the same server rack and two copies in a different rack. This helps with availability, so that even if one copy of the data faces an issue, the other two copies will be available. Even though HDFS has a built-in fault tolerance mechanism, it will not be able to foresee human errors such as copying or deleting data. As a precaution, additional backups at a different location may be required.

Metadata information is used to store information about a file for easy retrieval and processing. An illustration of data stored in different DataNodes is shown in the following figure:

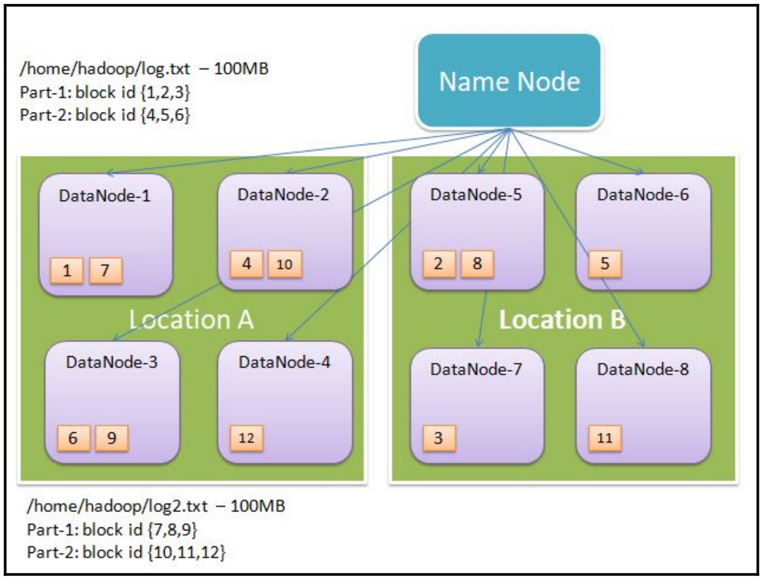


Figure-3.1.2

In the previous figure, two files, `log1.txt` and `log2.txt`, are stored in the HDFS. They are both 100 MB. Now, as the default block size in HDFS is 64 MB, these files will be split into two parts and, as per default configuration, each data block will be replicated to three data nodes. NameNode will assign a block ID to each data block. This metadata information will be stored in NameNode itself and the actual data block will be forwarded to DataNodes for storage purposes. The other important thing to notice here is that HDFS stores at least one copy of a data block in a different location; this may be on entirely different premises or just in a different network rack.

Apart from metadata management, NameNode is also responsible for health reports for each DataNode. This is done using a heartbeat. DataNode sends a heartbeat at a regular interval to notify NameNode that it is alive. The default interval time is 3 seconds. If any data node fails to send a heartbeat, NameNode will wait for up to 10 missed heartbeats, 30 seconds, before acting. NameNode will then generate an alert to replace and mark that DataNode as dead until it is replaced or returns. If any future data block retrieval happens that involving a dead node, NameNode will point it to the next available copy of that data block.

NameNode also facilitates a load balancing mechanism, so that if any DataNode is already heavily loaded, it will assign a task to the next available DataNode that contains required data blocks. This will help with the faster processing of a large dataset and utilizing the maximum throughput of available servers.

HDFS hands-on

Now let's get hands-on with HDFS. We will run the following tutorial on the virtual machine we set up in `Chapter 2`, *Big Data Environment Setup*. Let's first start with some basic commands, which are similar to Linux. All Hadoop commands require the `hdfs` prefix. If the command is related to a file system operation such as directory listing, creating, modifying, and deleting files and directories, it requires the `dfs` operator immediately after the `hdfs` prefix. Let's now execute our first command in Hadoop.

Creating a directory in HDFS

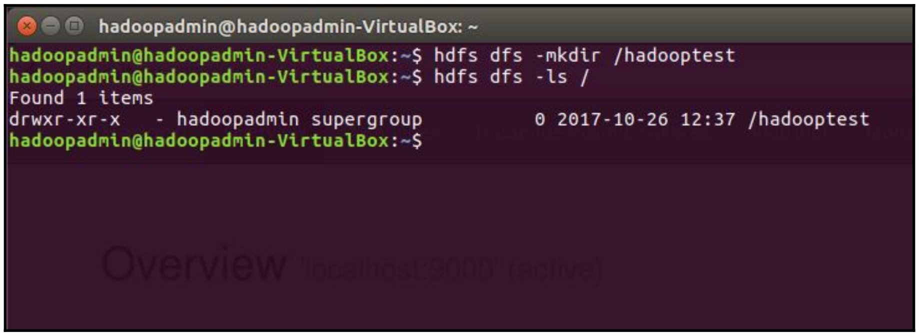
The following command will create a directory called hdfs dfs -ls / `hadooptest` in the root of the HDFS file system:

```
$ hdfs dfs -mkdir /hadooptest
```

Now execute the following command to list the files and directories present in HDFS, and to verify that the previous command executed successfully and has created a directory for us:

```
$ hdfs dfs -ls /
```

The preceding command lists the files and directories present in the Hadoop file system. The output will be similar to the following figure. The following screenshot shows only one directory, `hadooptest`, which we have just created. No other file and directory is present as it's a fresh installation:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -mkdir /hadooptest
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoopadmin supergroup          0 2017-10-26 12:37 /hadooptest
hadoopadmin@hadoopadmin-VirtualBox:~$
```

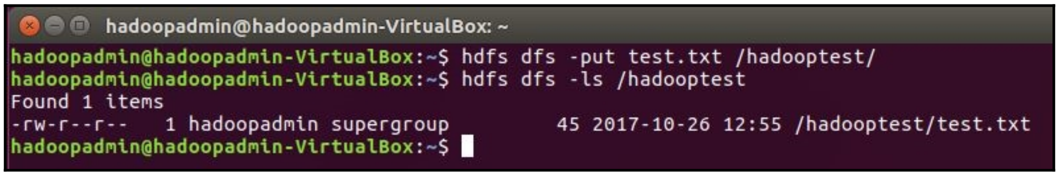
Figure-3.2.1

Copying files from a local file system to HDFS

We have created a file called `test.txt` in our home directory. To copy this file from the home directory, that is, the local file system storage, to HDFS inside the folder `hadooptest`, we need to execute the following command. The operator required for this purpose is `-put`:

```
$ hdfs dfs -put test.txt /hadooptest/
```

To test whether the file has been copied successfully, execute the `ls` command, as we have done previously. The output should be similar to the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -put test.txt /hadooptest/
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -ls /hadooptest
Found 1 items
-rw-r--r--   1 hadoopadmin supergroup         45 2017-10-26 12:55 /hadooptest/test.txt
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Figure-3.2.2

If you don't specify the destination folder when putting a file in HDFS, it will copy the file to a similar directory structure as the local file system if the same directory structure is present in HDFS.

Copying files from HDFS to a local file system

Now we will try to retrieve a file we put in HDFS, `test.txt`. The operator required for this purpose is `-get`. For this exercise, create a folder, `hdfsfiles`, in your local file system by executing the following command:

```
$ mkdir hdfsfiles
```

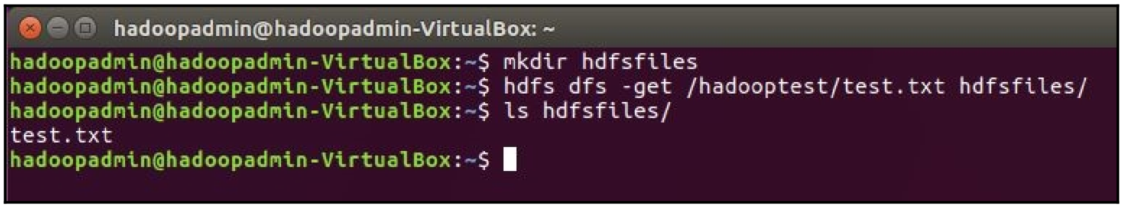
After creating the folder, we will extract the file from HDFS and put it in the local system by executing the following command:

```
$ hdfs dfs -get /hadooptest/test.txt hdfsfiles/
```

To verify that the file has been copied to our local file system, execute the following command:

```
$ ls hdfsfiles/
```

The output of these commands should be similar to the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ mkdir hdfsfiles
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -get /hadooptest/test.txt hdfsfiles/
hadoopadmin@hadoopadmin-VirtualBox:~$ ls hdfsfiles/
test.txt
hadoopadmin@hadoopadmin-VirtualBox:~$ 
```

Figure-3.2.3

Deleting files and folders in HDFS

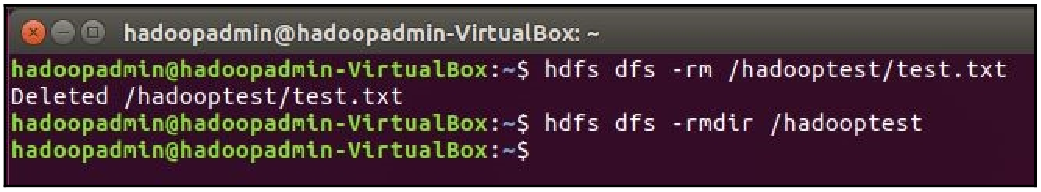
In the previous example, we copied the `test.txt` file in the `hadooptest` folder. Let's now try to delete this file. To delete it from HDFS, execute the following command:

```
$ hdfs dfs -rm /hadooptest/test.txt
```

Now we will try to delete the folder `hadooptest`, which we created in our first example of working with Hadoop hands-on. To delete this folder from HDFS, execute the following command:

```
$ hdfs dfs -rmdir /hadooptest
```

The output of the preceding command should be similar to the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -rm /hadooptest/test.txt
Deleted /hadooptest/test.txt
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -rmdir /hadooptest
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Figure-3.2.4

Hadoop MapReduce

Now that we have an understanding of how HDFS works, it's time to move forward and seek to understand what the benefit of HDFS and a clustered computing environment is. Hadoop introduces the MapReduce framework to facilitate the execution of programs and parallel processing. The following figure illustrates where the MapReduce framework fits into Hadoop's architecture:

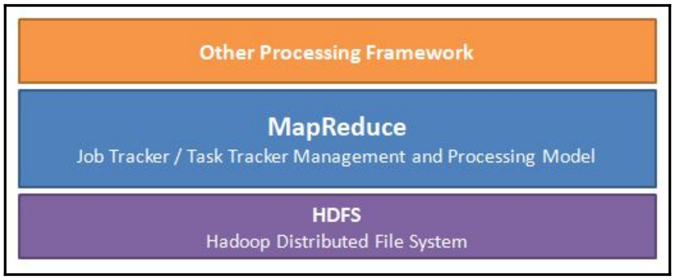

Other Processing Framework
MapReduce
Job Tracker / Task Tracker Management and Processing Model
HDFS
Hadoop Distributed File System

Figure-3.2.5

This framework mainly consists of two parts: **Map** and **Reduce**. The `Map` process mainly comprises of getting information from the data stored, applying the required algorithm, and generating a result in the form of key-value pairs. The `Reduce` process is used to summarize the information that was collected and paired during the `Map` process. We will look at these processes in more detail as we proceed with this chapter, but let's first understand how a program is executed in Hadoop.

Job Tracker and Task Tracker

Similar to HDFS, there are two main processes involved in the execution of any program, which also works using the Master Slave concept. They are Job Tracker and Task Tracker. Job Tracker is the master process, which receives requests for any program execution from a client. It then gets related metadata information from NameNode and assigns it to Task Tracker for further execution. The following figure illustrates the overall architecture including Job Tracker and Task Tracker:

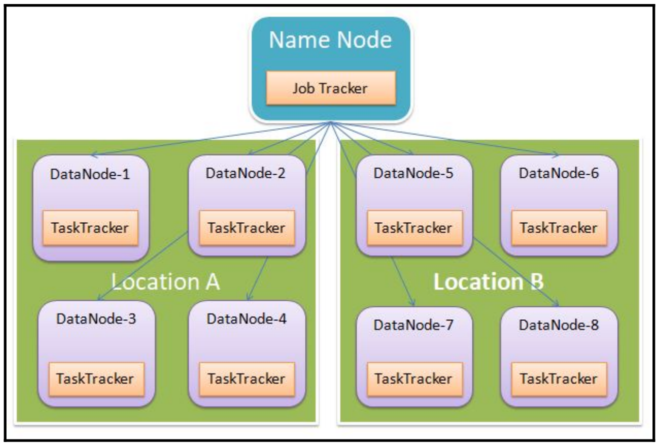


Figure-3.3.1

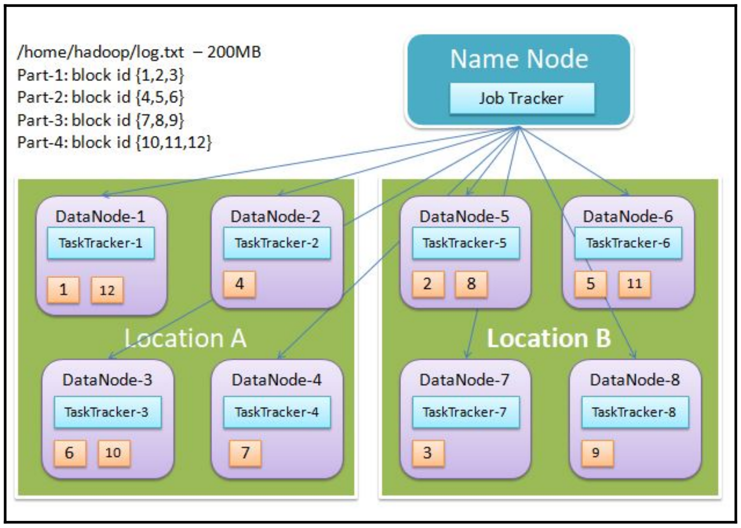
Job Tracker is a critical process as it manages the information of all of the program executions in the full Hadoop cluster. This is why this process resides in NameNode. Job Trackers also manages Task Tracker's activity and receives regular updates from it when it is operational and executing a task. As part of the default configuration, Task Tracker sends a heartbeat every 3 seconds, and Job Tracker will wait up to 30 seconds, 10 heartbeats, before marking that Task Tracker as dead. It will then reassign the same task to other data node that contains the same data.

Job Tracker is also responsible for load balancing between different data nodes. For example, if any data node is already processing a program and a new program request to access that data exists, Job Tracker will assign the new program to its replicated data node to avoid overloading and to benefit from parallel processing.

The execution flow of MapReduce

In a traditional system, when a client program executes, it requests data that will then be made available in the client's system for further processing. In the Hadoop system, the program will be sent towards data so that if data resides in multiple data nodes, the copies of a program will be sent to each data node, making it process in parallel. This is where the MapReduce framework comes in, as it pictures how a program will execute, what the flow will be, and how it will generate a result.

Job Tracker assigns tasks to Task Tracker before it starts processing on the data node. Let's suppose we have a file named `log.txt` that is 200 MB. As per the default configuration, it will be split into four data blocks; three of them will be 64 MB and the fourth one will be 8 MB. The following figure illustrates the storage of this file in different data nodes:

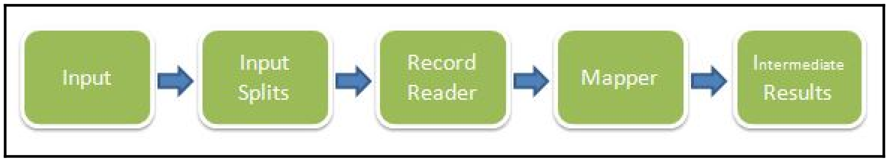


The input and output of a MapReduce program is written on the file system.

Mapper

Mapper is a process of generating results in the form of key-value pairs based on the input provided. It takes input as `<key, value>` pairs and outputs them in the same format that can be different in terms of data type.

The number in Mapper depends on the number of Input Splits involved in storing data. Let's assume that there are four Input Splits involved - there will four mappers involved in processing. The following figure illustrates what happens in the Mapper function:

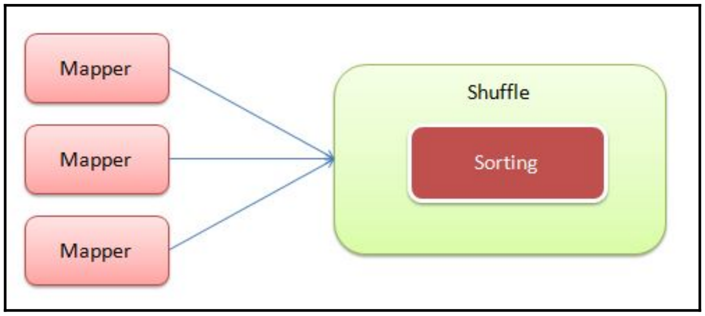


Record Reader is responsible for converting data into key-value pairs for Mapper. When the program executes, Mapper will run on DataNode-1, DataNode-2, DataNode-3, and DataNode-4 to take advantage of parallel processing; this is because independent data nodes contain different data blocks of the same file. Mapper will then produce output in the form of key-value pairs. This result is called Intermediate Results.

Shuffle and Sort

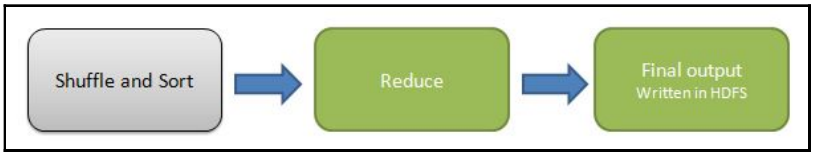
Between the Map and Reduce steps, there is a step called Shuffle. It is a process of transforming results from the Mapper phase and making them available to the `Reducer` function. All Mappers were executed independently on different data nodes and produced results as Key Value pairs. The shuffling step will not wait for Mappers to finish their jobs and will start processing them as they move along. Whichever Mapper finishes the job, Shuffle will start to transform its data.

The key-value pairs of the intermediate result will be sorted before passed to Reducer. This sort is done only on keys and not on values. Values will be passed to Reducer without sorting. The following figure illustrates this process:



Reducer

After Shuffle, the intermediate output will then be passed on to Reducer as an input. Reducer mainly performs an aggregation function on the key-value pairs for various calculations depending on the program's purpose. The aggregation can be of any type, including summation, multiplication, and so on, as illustrated in the following figure:



Finally, the program output is then written to the HDFS file system. It is worth noting that the number of reducers is configurable in the Hadoop system; by default there is one.

Example program

It's time to get our hands dirty and write our first program in Hadoop using the MapReduce framework. We will be using Java as a programming language for this purpose. The objective of this program is to find the total amount paid for each utility bill. For this purpose, we will create a text file that contains the required information. The following step-by-step tutorial will help us achieve this.

Preparing the data file for analysis

Create a text file in your file system and enter some dummy that can be utilized in our calculation. The format for entering text in the file is as follows:

```
Utility Name<space>Year-Month<space>Amount.
```

The following is an example of our text file. Alternatively, you can download a text file from the GitHub repository, by visiting `https://github.com/smfahad/BDAH-chapter3-example-data.git`:

```
electricity 2017-01 2000
gas 2017-01 700
telephone 2017-01 1150
electricity 2017-02 2230
gas 2017-02 850
telephone 2017-02 1350
...
...
```

Log.txt file sample data

Now create the `utility` directory and copy this file into HDFS by executing the following commands:

```
$ hdfs dfs -mkdir /utility/
$ hdfs dfs -put log.txt /utility/
```

Program code

Now it's time to write a Java program to be executed in Hadoop. This Java program will have three parts: the driver code, the Mapper code, and the Reducer code. The driver code contains the main method and the parameters needed to process a MapReduce job. The Mapper code contains how individual Mappers performs its calculation and the Reducer code contains how the results will be combined. We will discuss these topics in detail as we continue.

The MapReduce framework can only work on Java Object types and not on Java Primitive types. This means there must be a class to access primitive types as a collection.

Driver program

The driver program will utilize the MapReduce framework in the Hadoop environment to calculate the total amount paid for each bill. It is the primary executioner as it contains the main method. It defines all of the parameters required by the Hadoop framework to process a program. Following is the complete code for our driver class; note that comments have been added to make the code self-explanatory:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class UtilityBillSummary {
  public static void main(String[] args) throws Exception {

      //Initiate Hadoop Configuration class
      Configuration conf = new Configuration();

      // It is to configure a MapReduce Job and assigning it a name
      // for logging purposes
      Job job = Job.getInstance(conf, "UtilityBillSummary");
      job.setJarByClass(UtilityBillSummary.class);

      // In this step we will define what are the names of out Mapper
      // and Reducer classes
      job.setMapperClass(UtilityMapper.class);
      job.setReducerClass(UtilityReducer.class);

      // Here we are defining the types of <Key, Value> pairs
      // Key will be of Text type i.e. String
      // Value will be of Long type
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(LongWritable.class);

      // Here we are assigning the input and output files
      // which will be passed while executing program
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      // Wait until MapReduce Job finishes
      System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Mapper program

This is the Mapper part of the main program that generates Key Value pairs of the input split provided by Record Reader. It extends the Hadoop Mapper class, specifying the types of data that will be used while processing. The first two types, *Object* and *Text,* are required when reading the text file. The line number offset will become the key and the actual text will become the value. The last two types are an intermediate result generated by Mapper. The `text` type will be the key and `LongWritable` will become the value of the generated key-value pairs. Following is the complete code for the Mapper class:

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class UtilityMapper extends Mapper<Object, Text, Text, LongWritable>
{

  // Initialize key and value part of the pair
  private Text utility = new Text();
  private LongWritable amount = new LongWritable(1);

  public void map(Object key, Text value, Context context)
                  throws IOException, InterruptedException {

    // Mapper receive line by line of the text file
    String row = value.toString();

    // Check whether the text line contains data or its empty
    if (!row.isEmpty()) {

      // Split the data based in <space> deliminator
      String[] rowValues = row.split(" ");

      // set name of the utility
      utility.set(rowValues[0]);

      // set amount paid for that utility
      amount.set(Long.parseLong(rowValues[2]));

      // assign Utility as Key and Amount as value in keyValue pair
      context.write(utility, amount);
    }
  }
}
```

Reducer program

This class extends the Reducer class, which contains generic types. To extend it, we need the types to be defined in a similar way to the process required in the Mapper class. We set up our Mapper to produce `<Text,LongWritable>` values that will be used as input in the Reducer class. These two types will become the first two values while extending the Reducer class. The last two values define the type of the final output key-value pairs. In our case, it will also be `<Text,LongWritable>`, as we have to produce a result that calculates the total amount against each bill paid. Following is the complete code for the Reducer class:

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class UtilityReducer extends
Reducer<Text,LongWritable,Text,LongWritable> {
  private LongWritable result = new LongWritable();

  public void reduce(Text key, Iterable<LongWritable> values, Context
context)
        throws IOException, InterruptedException {
          // Initiate variable total
          long total = 0;

          // Go through all the values for a specific key and sum them up
          // to achieve our desired output.
          for (LongWritable val : values) {
                total += val.get();
          }

          // Assign calculated value
          result.set(total);

          // Assign the Utility as Key and the summed amount as value
          context.write(key, result);
    }
}
```

Observations and results

It's now time to execute your program in the Hadoop environment. If you are using Eclipse or any other IDE to write your code, you can just export your solution as a JAR file. If you are using a text editor to write your code, you need to compile it and make a JAR file from the compiled class files. Put all three classes in the same folder and execute the following command to compile them:

```
$ javac -classpath $(hadoop classpath) UtilityBillSummary.java
UtilityMapper.java UtilityReducer.java
```

Once that has been complied successfully, execute the following command to make a JAR file that contains the compiled `.class` files:

```
$ jar -cvf utilitysummary.jar *.class
```

Now it's time to pass your JAR file to Hadoop for execution with the following command:

```
hadoop<space>jar<space>{yourfilename.jar}<space>{MainClassName}<space>{inpu
tfilename.txt}<space>{outputDirectory}
```

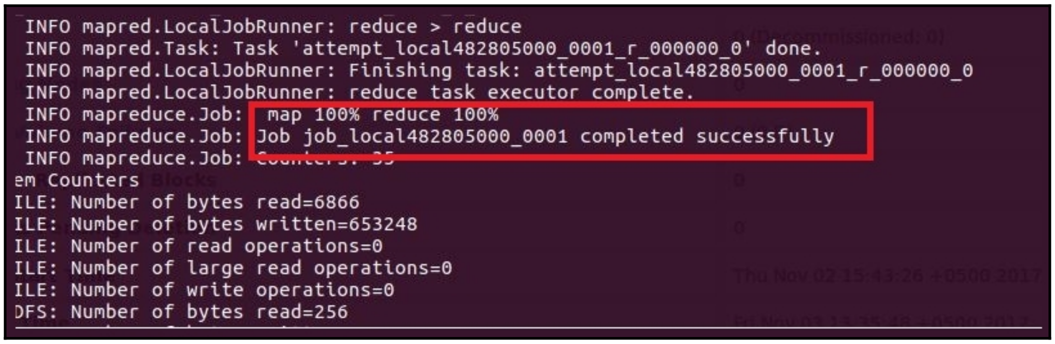
In our case, the values will be as follows:

Command fields	**Values**	**Comments**
JAR file name	`utilitysummary.jar`	The JAR file that we compiled previously
Main class name	`UtilitySummary`	Our main driver class
Input file in HDFS	`/utility/log.txt`	The text file on which we want to perform our analysis
Output directory	`/utilityOutput`	The result will be published to this directory after completing the MapReduce job

Type in the following command as per the values mentioned previously to execute your first `MapReduce` Job:

```
$ hadoop jar utilitysummary.jar UtilitySummary /utility/log.txt
/utilityOutput
```

The following figure shows a few lines from the output generated in the terminal. This output contains the successful completion message:

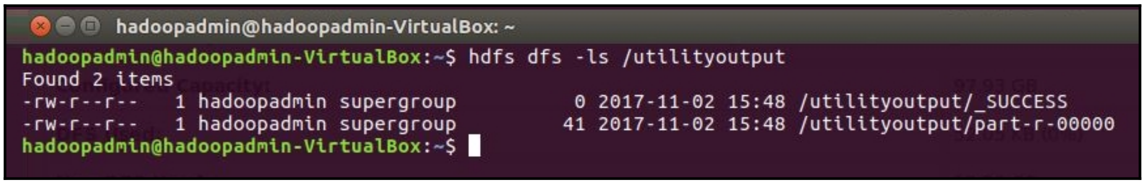


```
 INFO mapred.LocalJobRunner: reduce > reduce
 INFO mapred.Task: Task 'attempt_local482805000_0001_r_000000_0' done.
 INFO mapred.LocalJobRunner: Finishing task: attempt_local482805000_0001_r_000000_0
 INFO mapred.LocalJobRunner: reduce task executor complete.
 INFO mapreduce.Job:  map 100% reduce 100%
 INFO mapreduce.Job: Job job_local482805000_0001 completed successfully
 INFO mapreduce.Job: Counters: 35
em Counters
ILE: Number of bytes read=6866
ILE: Number of bytes written=653248
ILE: Number of read operations=0
ILE: Number of large read operations=0
ILE: Number of write operations=0
DFS: Number of bytes read=256
```

Now when you browse the output directory `/utilityoutput`, you should see the two following files:

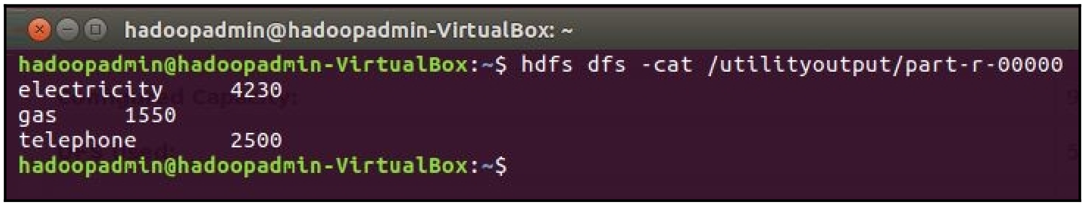
- `_SUCCESS`, which tells you that the job was completed successfully
- `part-r-00000`, which contains the actual result gathered; there may be multiple files here, with a sequence, such as `00001`, `00002`, and so on

The following figure displays the files generated in the output directory:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -ls /utilityoutput
Found 2 items
-rw-r--r--   1 hadoopadmin supergroup          0 2017-11-02 15:48 /utilityoutput/_SUCCESS
-rw-r--r--   1 hadoopadmin supergroup         41 2017-11-02 15:48 /utilityoutput/part-r-00000
hadoopadmin@hadoopadmin-VirtualBox:~$
```

To view the result of your program, display the contents of the `part-r-00000` file as shown in the following screenshot. As you can see, it displays the total amount paid for each utility:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -cat /utilityoutput/part-r-00000
electricity     4230
gas     1550
telephone       2500
hadoopadmin@hadoopadmin-VirtualBox:~$
```

YARN

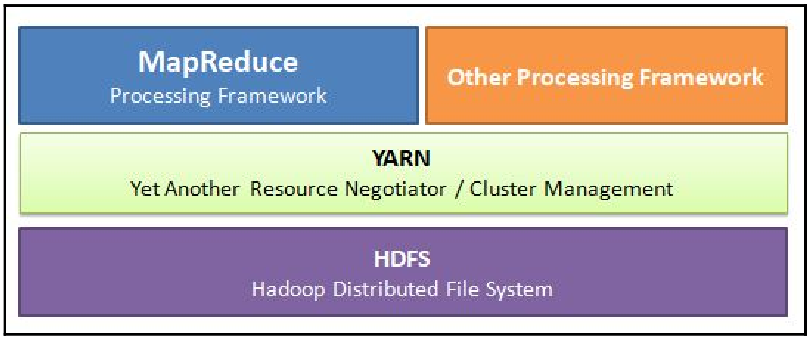
Now we have completed some hands-on experience using the MapReduce framework. You should now be able to recall that in MapReduce, we see a service called **Job Tracker**. Job Tracker does the following things:

- It gets the information from NameNode to figure out where the data that will be required by the client program resides
- It coordinates with respective Task Trackers to assign jobs
- It monitors the life cycle of a Task Tracker's activities

In a normal scenario, it seems perfectly acceptable that a single machine can handle the preceding activities. We should instead be thinking of these services taking place in a production environment with a large cluster, where thousands of jobs will be running. This means that the Job Tracker will be performing several activities, a burden for the Job Tracker and an inelegant solution. This way of working can also lead to a performance bottleneck, as new jobs will continuously arrive to a single point of contact in Hadoop.

The second problem in the current Hadoop framework is resource allocation. So, let's assume that we have four machines in a cluster. Two of them have 8 GB of RAM and the other two have 16 GB of RAM with an equal amount of storage capacity. When Job Tracker assigns a job to Task Tracker, Task Tracker is only responsible for processing that task within its own machine resources. It is also worth mentioning that in this scenario, all machines will be considered the same and Task Tracker will performs a task whether it has 8GB of RAM or 16GB available.

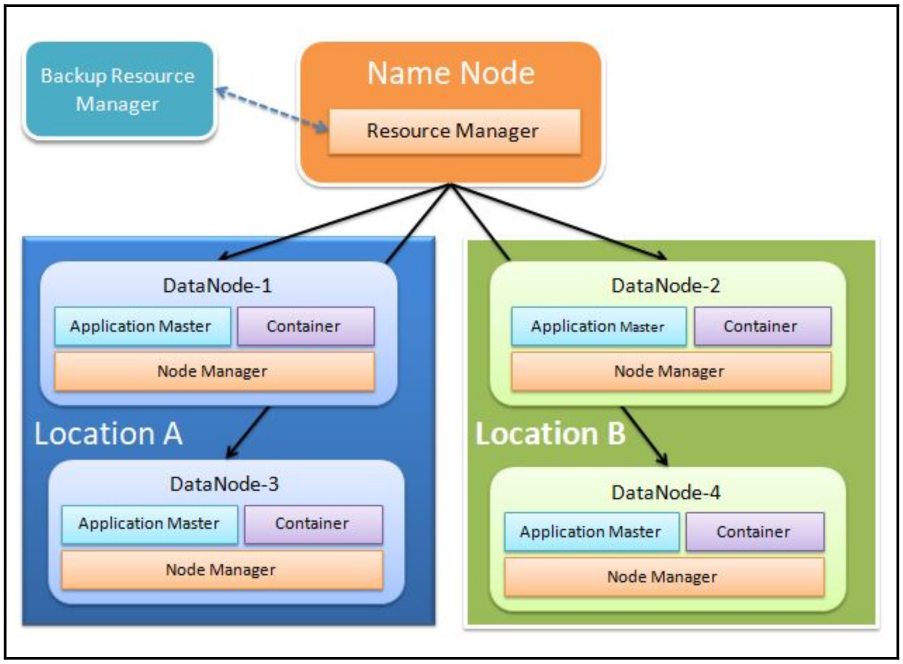
To overcome these architectural issues, Hadoop has introduced a new processing model called YARN (Yet Another Resource Negotiator) to replace the Job Tracker and Task Tracker model. It is also called Hadoop 2. The following figure illustrates Hadoop's architectural with YARN included:


MapReduce
Processing Framework
Other Processing Framework
YARN
Yet Another Resource Negotiator / Cluster Management
HDFS
Hadoop Distributed File System

YARN divides the functionality of a Job Tracker into different services. It works with the following four new services:

- Resource Manager
- Node Manager
- Container
- Application Manager

The following figure illustrates YARN's model and where the preceding services will run:



YARN supports programs written in older APIs, that is, MapReduce v-1, as well programs written in newer APIs.

Resource Manager

We know that YARN has introduced a new cluster management architecture. One of the new components of this architecture is Resource Manager. Resource Manager has somewhat limited functionality compared to Job Tracker. In the previous model, Job tracker was responsible for taking care of all the activity and its status in the cluster, whereas Resource Manager is responsible for the following:

- Assigning an ID to a new job submitted by a client
- Initiating a new daemon service called Application Master on the available Node Manager and assigning the job to this Application Master for further execution and handling
- Managing and assigning resources across clusters in the form of containers.

In Hadoop V1, we had only one Job Tracker. In case of any failure in Job Tracker, the whole cluster execution will collapse. Now, with Hadoop v2, there is also a backup Resource Manager that can replace the original in the case of any failure.

Node Manager

Node Manager is a service daemon that runs on all data nodes. It means that every data node has its own Node Manager. It is responsible for monitoring resources, such as memory, CPU, and storage and reporting back to Resource Manager how much is available and how much has been utilized.

Container

A container is a slot representation of CPU and memory utilization that has been assigned by Resource Manager in a data note in any cluster. Containers are managed by Node Manager. A container is a place where a program resides and executes its tasks, such as `Map` and `Reduce`.

Application Master

Once a client submits a new job to Resource Manager, it will then contact any available Node Manager and initiate the Application Master service in the data node. Application Master will then contact NameNode to retrieve the metadata required to execute the job. It will then create one `Map` task per input split.

Application Master will also decide how many containers it requires to execute a job. If the task is small, Application Master executes them within its own JVM or container, which helps to reduce the overhead of requesting and allocating a separate container. These small tasks are commonly known as **Uber Tasks.**

Parameters of Uber Jobs can be configured by editing the `mapred-site.xml` file in `/etc/hadoop/ directory` of the Hadoop installation. These parameters are as follows:

Variables	**Configurable values**	**Comments**
`mapreduce.job.ubertask.enable`	True or false	Enables or disables the handling of Uber Tasks
`mapreduce.job.ubertask.maxmaps`	Any number	Defines the maximum number of Mappers a job can contain before considered an Uber Job
`mapreduce.job.ubertask.maxreduces`	Any number	Defines the maximum number of Reducers a job can contain before considered an Uber Job
`mapreduce.job.ubertask.maxbytes`		Defines the maximum byte size of a job before considered an Uber Job; if no value is specified, the default block size in the HDFS configuration will be considered the maximum number of bytes

Application Master will terminate itself after completing a job assigned to it. You may think of Application Master as a person hired for a specific task who leaves once they've finished.

Apache Projects related to big data

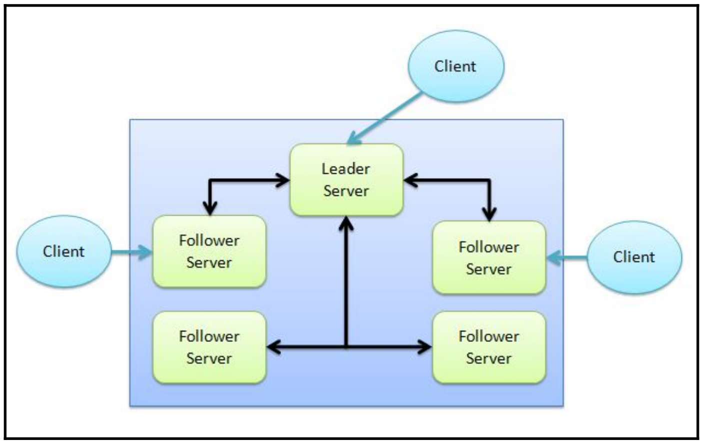
We have now developed an understanding of how a program works and is executed in the Hadoop environment, how resources are allocated to a program, and Hadoop's approach to stored data. This information is mostly related to cluster management and custom application processing, but there are many Apache Projects that are related to big data processing that will help to process data. We will now briefly go through some of these projects, as we will be using some of them in our examples and programs later on in this book.

Apache Zookeeper

Zookeeper is an open source Apache Project that provides a centralized infrastructure and services that enable synchronization across a cluster. It is responsible for maintaining synchronization between different objects such as configuration files, hierarchical namespaces, messaging, notifications, and so on. The Zookeeper framework is mainly designed for use in setup synchronization within a distributed clustered environment. Imagine that you need to build an application for a distributed environment for 10 to 20 servers that requires synchronization services for data maintenance. Despite such a small setup, you will have to write your own services and confront any issues arising from synchronization. In this situation, Zookeeper would be a great support. Zookeeper is also designed to be very fast and work well with heavy loads in a cluster environment.

Zookeeper is written in Java. It exposes its API to be used by the application in Java as well as in C-programming languages.

Now we will go through the architecture of Zookeeper. Zookeeper uses the leader and followers concept, where consistency in data relies on a leader that can provide and verify data. The following Figure illustrates the Zookeeper Quorum configuration:



It is recommended that Zookeeper Quorum has an odd number of nodes, as some features in Zookeeper require a voting mechanism. For example, if you have five nodes in Quorum, three votes are needed as a majority to elect a leader. If you have six nodes in Quorum, you need four votes to win a majority, the same needed if you have seven nodes. One extra node will make your life easier without much overhead.

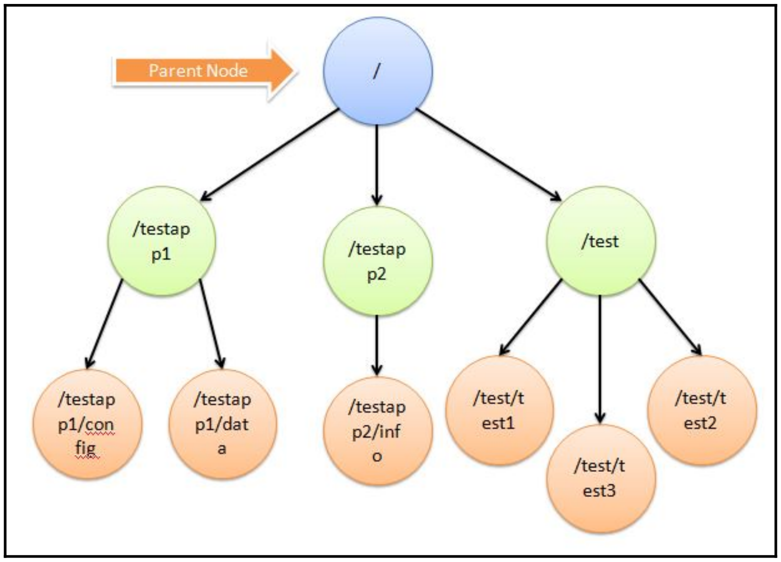
Whenever a leader is unable to get a majority because of hardware failure, network failure, or any other reason, it steps down and the Zookeeper system becomes read-only. Similarly, when the leader itself fails, the remaining nodes will elect a new leader.

Zookeeper is a fault-tolerant system. It makes sure that replication has been done properly. So, in the case of a write operation, a request first goes to the leader before being forwarded to the available nodes. If the majority of nodes receive the updated data and write it persistently, Zookeeper will not commit and confirms to the client that the data has been successfully written. This all happens in parallel.

TIP

The Zookeeper API contains two definitions for every function. One function is for synchronous calls and the other one is for asynchronous calls. You can select any one of them based on your requirements.

Zookeeper works in a tree-like directory structure. Each node in a tree is called a **znode**. The top-most znode in the hierarchy is called the **Parent node**, which is also represented by `/`. Most applications that use Zookeeper create their node after the parent node. The most important thing to note here is that each node can contain data as well as its child node. The following figure illustrates the directory structure of Zookeeper and how it stores information:



There are three types of znode; they are as follows:

1. **Persistent Node**: A type of node that once created, will remain stored until it is deleted deliberately by the application or user.
2. **Ephemeral Node**: A type of node that once created, will remain there until the client is connected and will be deleted once the connection is closed.
3. **Sequential Node**: A type of node that once created, Zookeeper attaches a 10 digit sequence number to, along with a prefix and a `create` command. It can be persistent or ephemeral in nature.

To sum up the operations of the Zookeeper, it provides highly reliable services while keeping many copies of the data and configuration. Its best use is to only store configuration files and manage the metadata information of your data. In Hadoop v1, Namenode was the single point of failure. Because of which many companies were afraid to use it as if the NameNode goes down, all of their data will be lost. To overcome this factor, in Hadoop v2, Apache introduces Zookeeper to handle NameNode failure. So that if active NameNode goes down, the standby node will take its place, and because of the zookeeper super fast performance, all this happen in a very minimal time leaving no to very little discontinuation in performance.

Apache Kafka

We have now learned how Hadoop clusters work and how they process data. We've so far assumed that data is already available in the cluster for analysis, but have you thought about how it was made available in Hadoop? How you will handle the stream of data coming in, for example? Will some sources continue to generate data that you have to bring into Hadoop in real-time for analysis? Consider that you are required to gather data from multiple sources. For that you need a mechanism that can handle multiple sources in a fault-tolerant manner.

Apache Kafka is real-time feed handling system that can receive data from multiple sources called **Producers** and that can provide data to multiple destinations called **Consumers** in a sequential manner. The following figure illustrates the flow of data in the Kafka framework:

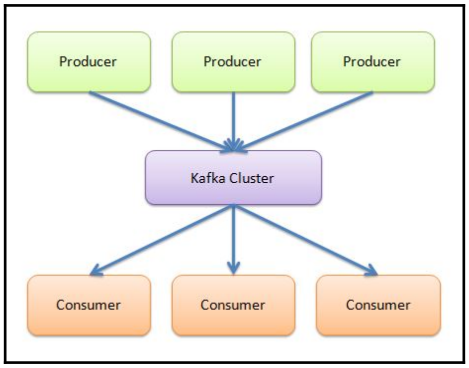
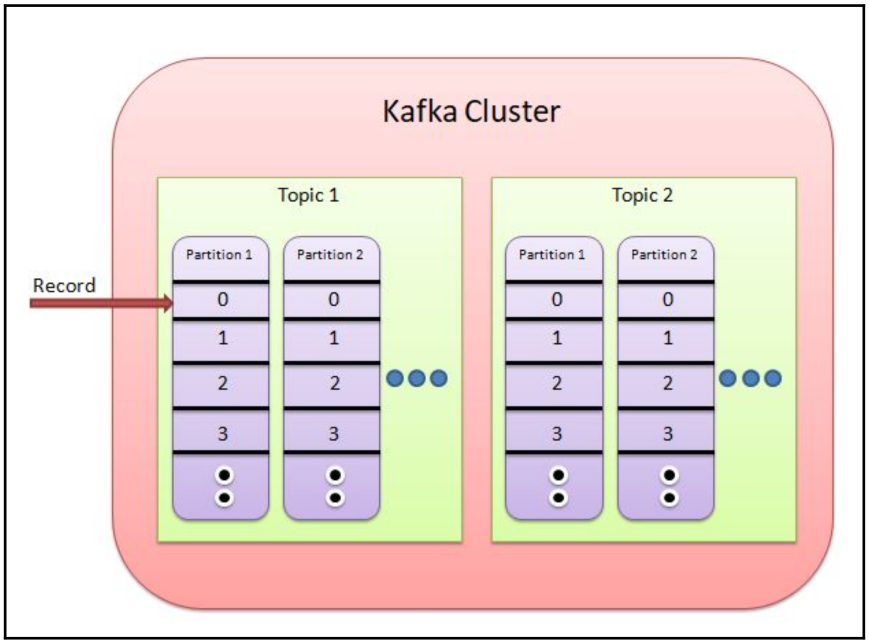


Figure-3.6.1

Each stream of data coming in and out from a Kafka cluster is known as a **Topic**. A topic labels the data that is coming from different producers. It also helps consumers request a particular dataset from a Kafka cluster in the form of a required topic. The following figure illustrates the internal mechanism of a Kafka cluster:



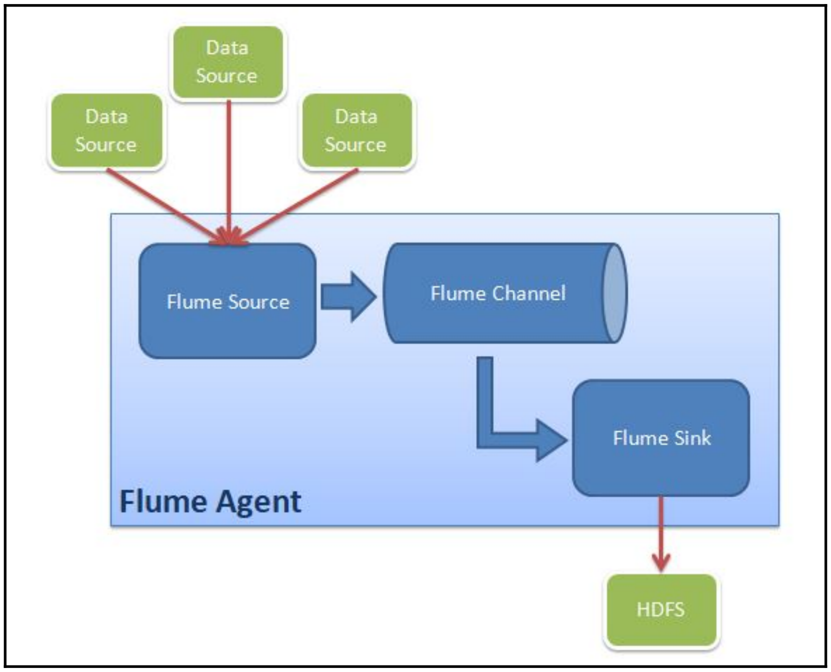
The topic itself is comprised of different partitions. These partitions are sequences of data blocks called **Records**. A producer submits a record to a broker in a Kafka cluster. The broker is then responsible for replicating that data in different partitions and maintaining its sequence. When a consumer requests data related to a specific topic, the broker is the one who is responsible for handling that request and providing the desired output in a sequential manner. For the broker, all client handling is done independently, so it doesn't affect one client if another acquires data slowly. For example, if one client received record 5 and is ready to receive record 6, at the same time broker can send record 3 to a different client.

One thing to keep in regarding Kafka is that it doesn't allow you to make any amendment once data has been sent a cluster for handling. This means that you will not be able to any updates of any sort nor be able to delete a record. Kafka doesn't perform much processing in memory; therefore, all records received by Kafka are directly written to the hard disk, making these records persistent.

The Apache Kafka framework is not just for Hadoop; it can be utilized in any input and output stream handling scenarios. In Hadoop, with the help of Kafka, we can analyze the stream of data as it comes in, instead of collecting the data first and then running batch jobs.

Apache Flume

Apache Flume is a framework that helps move a large amount of streaming data from one place to another. It is primarily designed for log collection and aggregation, obtaining data across different servers before moving it to a centralized place such as Hadoop for processing and analysis. Its usages not limited to just log aggregation, though. Flume's data source connectors are customizable and can be used to transport a large amount of event-generated data such as network traffic data, social media data, and so on. It is a fault-tolerant system with many failure and recovery mechanisms. The following figure illustrates how Apache Flume works:



There are three main components inside a Flume agent. They are as follows:

- **Flume source**: External Data sources including web servers or social media generate a log or event-related data; these events will be consumed by a flume source. For example, if a Flume source is configured to monitor a Twitter account, it will generate an event for every tweet and pass it on. Once a Flume source receives the event, it will pass it on to a Flume channel. It has no control over how a Flume channel will store or treat the data.
- **Flume channel**: A Flume channel is connected to both a Flume source and Flume sink. Once any event reaches a Flume source, it will be immediately passed on to a Flume channel, which acts as a buffer once you have configured how much data it can store. A Flume channel is responsible for storing data until it is consumed by the Flume sink. It may use a local file system to store data for more reliability.
- **Flume sink**: A Flume sink is responsible for retrieving an event from a Flume channel and passing it on to an external repository, such as HDFS. The event will be removed from the Flume channel once it is consumed by a Flume sink.

Flume is a framework that helps with data handling while streaming it in to Hadoop. It is often compared to Apache Kafka, although they are two quite different systems. Some of their functionalities may overlap, but that doesn't mean Apache Kafka is an improved version of Apache Flume or vice versa. Kafka is more of a general-purpose system that can be used in messaging or in other types of systems that require an approach with a fault-tolerant mechanism. On the other hand, Flume is a reliable data handler tightly built for the Hadoop ecosystem that brings a large amount of log and streaming data into the Hadoop cluster. It provides many built-in sources and sinks, which can be used directly in your application.

Apache Cassandra

Cassandra is an open source Apache Project designed for NoSQL database management. It was initially designed by the Facebook team for their internal use, specifically to enhance their search engine mechanism for unstructured data. It was later released by Facebook as an Apache Project.

Cassandra runs on a distributed environment. Its architecture was designed with the idea that software and hardware fails in mind. So, if anything fails in a cluster, it doesn't mean that the whole operation also has to fail.

Some of the key features of Apache Cassandra are as follows:

- All nodes in a Cassandra cluster have the same role; in other words, there is no master node or leader node. Every node can service a client request. This makes it a system with no single point of failure and high availability.
- Cassandra runs across multiple nodes across multiple data centers. It replicates data across these data centers to avoid failure or downtime. This makes it a highly fault-tolerant system.
- Cassandra supports integration with Apache Hadoop. It also supports MapReduce.

Cassandra has a component called Gossip Protocol. The primary purpose of Gossip Protocol is that the nodes in a Cassandra cluster can communicate with each other internally as part of their core. By using such a protocol, nodes can share their current state, location, heartbeats, and much more. Information collected using Gossip is persisted. In case of any temporary failure or node restart, it will be immediately available for the node to pick up from where it left off.

The main brain behind replication in Cassandra is called Partitioner. Partitioner decides which node will get the first copy of data and how other nodes will receives remaining copies. It uses a primary key, which is also known as a partition key, to identify a unique row. An interesting point to note is that a partition key is not just used to classify the uniqueness of data but it also contains other related information. The most important information it contains is about data locality, meaning it has the address of the nodes where the data resides.

Cassandra uses it own programming language to access data across its nodes. It is called Cassandra Query Language or CQL. It is similar to SQL, which is mainly used by Relational Databases. CQL can be used by running its own application called cqlsh. Cassandra also provides many integration interfaces for multiple programming languages to build an application using Cassandra. Its integration API supports Java, C++, Python, and others.

Apache Cassandra is a very popular, free, and open-source tool for unstructured data management with no single point of failure. We will discuss Apache Cassandra in detail in our next chapter, which is all about NoSQL, its related technology, and frameworks.

Apache HBase

HBase is another Apache Project designed to manage the NoSQL data store. It is designed to make use of Hadoop Ecosystem's features, including reliability, fault tolerance, and so on. It utilizes HDFS as a file system for storage purposes. There are multiple data models that NoSQL works with and Apache HBase belongs to the column-oriented data model. HBase was originally based on Google Big Table, which is also related to the column-oriented model for unstructured data.

HBase stores everything in the form of a key-value pair. The important thing to note is that in HBase, a key and a value are in the form of bytes. So, to store any information in HBase, you have to convert information into bytes. (In other words, its API doesn't accept any thing other then byte array.) Be careful with HBase, as when you store data, you should remember its original type. Data that was originally a string will return as a byte array if recalled incorrectly. As a result, it will create a bug in your application and crashes your application.

HBase is written in Java. It also exposes an API for Java integration. Oher programming languages have to use RESTful web services or the Thrift gateway.

HBase doesn't have any SQL tool built-in, so as an alternate, you have to use another application or third party tools to write SQL queries that directly run on HBase. Some of these projects include Apache Phoenix, Hive, and Presto.

When we think about the design consideration of an HBase database, we should not think of it as a relational database where you try to find a relationship between each object. If you think of it in this way, you will end up with a poorly-performing, un-optimized database. In relational database design, you first identify the relationship between different entities, build an index, and do performance tuning in order to get quicker results when you execute SQL queries. In the case of HBase, it is the total opposite. You have to build your queries first so as to identify how you will approach data. You have to normalize and denormalize your data based on which entities have been accessed together and not on the basis of their relationship.

Apache HBase is a free and open-source tool for unstructured data management. We will discuss it in detail in our next chapter, which is all about NoSQL database and its related frameworks.

Apache Spark

Apache Spark in an open-source framework designed for multiple purposes. It has different processing components, which are as follows:

- **Spark Core**: It is like the main brain of Spark. It handles the most critical functions within Spark, such as task scheduling.
- **Spark SQL**: It is designed to work with structured data. It can run SQL as well as HiveQL queries to seamlessly integrate Spark in an existing application. It also supports JDBC and ODBC connectors as per the industry standard to connect to Business Intelligence tools.
- **Spark Streaming**: It is designed to work with streams of data in a scalable and fault-tolerant manner. It splits streaming data into mini batches and then executes a job. It can also be linked with an existing data stream, like in HDFS, Kafka, and Flume. You can also create your own data source if needed.
- **MLib**: It is designed to execute a machine learning algorithm and perform statistical analysis on your big data with the super fast processing power of Spark. The list of algorithmic topics includes classification, regression, decision trees, random forests, and gradient-boosted trees. It also contains algorithms related to recommendation, clustering, topic modeling, frequent itemsets, association rules, and sequential pattern mining.

- **GraphX**: It is designed for graph-related analytics. It has a vast list of algorithms that have been contributed by its users.

The following figure illustrates Spark's components in Hadoop Ecosystem:

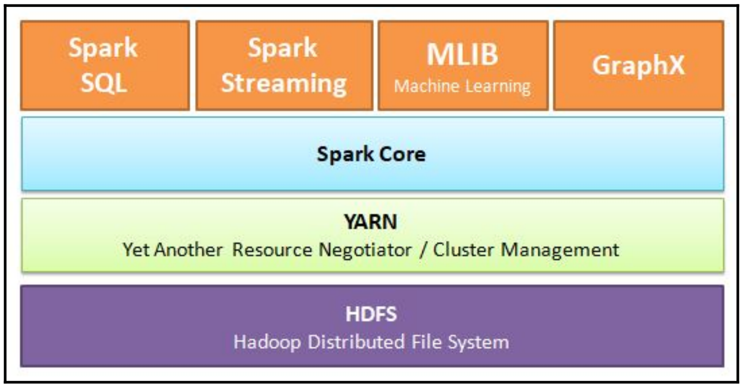

Spark SQL
Spark Streaming
MLIB
Machine Learning
GraphX
Spark Core
YARN
Yet Another Resource Negotiator / Cluster Management
HDFS
Hadoop Distributed File System

Figure-3.8.1

Apache Spark is one of the top choices for big data architects because of the components and libraries it supports. The application developer doesn't have to look for a different framework to do any of the listed functionalities, and some of its key features are as follows:

- Spark's main attraction is its memory processing of large datasets. Spark persists the data in memory and also on the disk if necessary. It also has an improved algorithm for processing data. As per the Spark's team benchmark, its response time is 100 times faster than MapReduce in memory processing and 10 times faster than Hadoop at disk operations.
- From a developer's perspective, Spark supports integration with Java, Python, Scala and R programming languages, while most frameworks only support Java.
- It supports multiple distributed processing frameworks including Hadoop YARN and Mesos, and in the cloud. It can also work in an independent cluster mode.
- It supports multiple data sources, including HDFS, Cassandra, HBase, or any Hadoop data source.

Apache Spark contains different libraries, as we have discussed, relating to big data processing. To understand Spark as a big data architect, you will need more than an overview. We will cover Spark's libraries in-depth in a different section of this book as we proceed.

Summary

In this chapter, we looked at various aspects of Apache Hadoop. We discussed the main components of Hadoop, such as the Hadoop Distributed File System, the MapReduce framework, and YARN. In between, we did some practical work by executing basic command related to HDFS. We also developed a program to calculate a bill summary using the MapReduce framework with easy-to-understand code.

Then, we discussed other projects under the umbrella of the Apache Foundation. These projects included Apache Zookeeper, Apache Kafka, Apache Flume, Apache Cassandra, Apache HBase, and Apache Spark. These projects are related to Hadoop Ecosystem. Some of them are related to bringing data into Hadoop, while others are related to the processing of data. The important thing we learned here is that though projects may appear similar, their uses and architecture differs. It's up to a big data architect to decide which framework best fits their required setup.

In the next chapter, we will look at NoSQL in detail, as well as its core concepts and principles. We will learn what the different data models in NoSQL are, as well as what applications and frameworks are available against each data model. We will also discuss the different designs available, helping you to choose the right model for you.

4
NoSQL Database

Nowadays, there is so much hype about NoSQL databases, especially in the big data world. People seem to be discussing different aspects of NoSQL and how they can get the most out of it. Different types of questions come to their minds, such as what is it? How it is different from RDBMS? How do I select an appropriate framework and tool while architecting my project?

In this chapter, we will go through NoSQL and answer all of these questions to build a strong foundation. We will then cover the following NoSQL databases from practical aspects, which includes their installation, basic configuration, and most of the operations that we normally perform in a database. We will be mainly discussing the following topics:

- What is NoSQL?
- Benefits of NoSQL
- Comparison of NoSQL and RDBMS
- CAP theorem and ACID properties
- Different data models in NoSQL
- Apache Cassandra
- MongoDB
- Neo4j

Let's start exploring the NoSQL world with a question: what is NoSQL?

What is NoSQL?

So far, we have the understanding that when we say the word **database**, the most common definition that comes to our mind is very well structured and formatted data stored in a tabular form. Now the question is, what will happen with a large amount of data that is unstructured and doesn't have the proper formatting or schema? What will we do then? Here is where NoSQL database comes into the picture. It is a mechanism for storing data that doesn't have any fixed schema. Most of the people assume that it means *No* SQL, whereas the actual abbreviation is from **Not Only SQL**. It means that it doesn't rely only on the SQL programming language for manipulating and storing data, but it can be used in conjunction with other programming languages. Now, we will discuss some of the benefits of NoSQL databases.

Benefits of NoSQL databases

NoSQL helps us to deal with data that we were not able to store or maintain using traditional system approaches. The following are the key benefits of NoSQL databases:

- **NoSQL provides schema less data storage** is one of the main advantages. It will allow the storing of all types of data in different formats and in different schemas, thereby providing more robust and agile development.
- **NoSQL servers scales horizontally**, which means it is very easy to scale the capacity up or down. Simply add new servers or remove servers to increase or decrease its capacity, storage, and computation power.
- **NoSQL works in a clustered environment** that is mainly built on commodity hardware, which is much less expensive than a highly reliable server without affecting the performance or reliability.
- **NoSQL databases spread across multiple nodes** with replication to multiple servers. Some of NoSQL database frameworks even work without the master slave concept, which makes them highly available with no single point of failure.

These are some of the key advantages of using NoSQL databases over the traditional approach. Now, moving forward, we will compare NoSQL with RDBMS databases to give you a clear understanding of the differences between both database types.

NoSQL versus RDBMS

We will now discuss the different characteristics of NoSQL and relational databases with a point by point comparison to give you a clear understanding:

RDBMS	**NoSQL**
RDBMS are relational databases.	NoSQL are normally non-relational databases or distributed databases.
RDBMS databases store data in tabular form, which mean it contains data rows and columns.	NoSQL databases are of key-value, document, column based, or graph based datastores.
RDBMS has predefined schema.	NoSQL have dynamic schema.
RDBMS is vertically scalable. It means you can only increase the hardware on the same server, increasing the computation or other hardware resources.	NoSQL database is horizontally scalable. It means you can add more servers in the cluster to increase the computation power and different hardware resources, such as storage and memory.
RDBMS relies on standard query language for all the related operations. Different RDBMS tools extend the features of the SQL by introducing their own advanced function to be utilized for handling databases.	NoSQL uses the basic format of the standard query language. It may differ from framework to framework, whichever you select to handle NoSQL database.
SQL supports complex and nested queries to extract the desired output.	NoSQL frameworks normally handle the basic CRUD operations as far as support to SQL is concerned. It doesn't have the interface to handle complex and nested queries.
SQL is more reliable in performing high transactional values.	NoSQL doesn't support high transactional operations.
SQL replies on **ACID** (**atomicity**, **consistency**, **isolation**, **durability**) properties. We will discuss ACID properties in detail as we proceed in this chapter.	NoSQL mainly follows the **CAP** (**consistency**, **availability**, **partition**) theorem. We will discuss the CAP theorem in detail as we proceed in this chapter.
SQL databases are classified based on open source or commercial application.	NoSQL classifies its different database frameworks based on the data model type it supports, such as key-value data stores, and column stores, and so on.

Example of RDBMS are MySQL, Microsoft SQL, Oracle, Postgres.	Example of NoSQL frameworks are Apache Cassandra, MongoDB, HBase, Redis, Neo4j.

SQL stands for **Standard Query Language**. It is a programming language used in **RDBMS** (**Relational Database Management System**) to store and retrieve data from a structured databases. It is as per the ANSI standard.

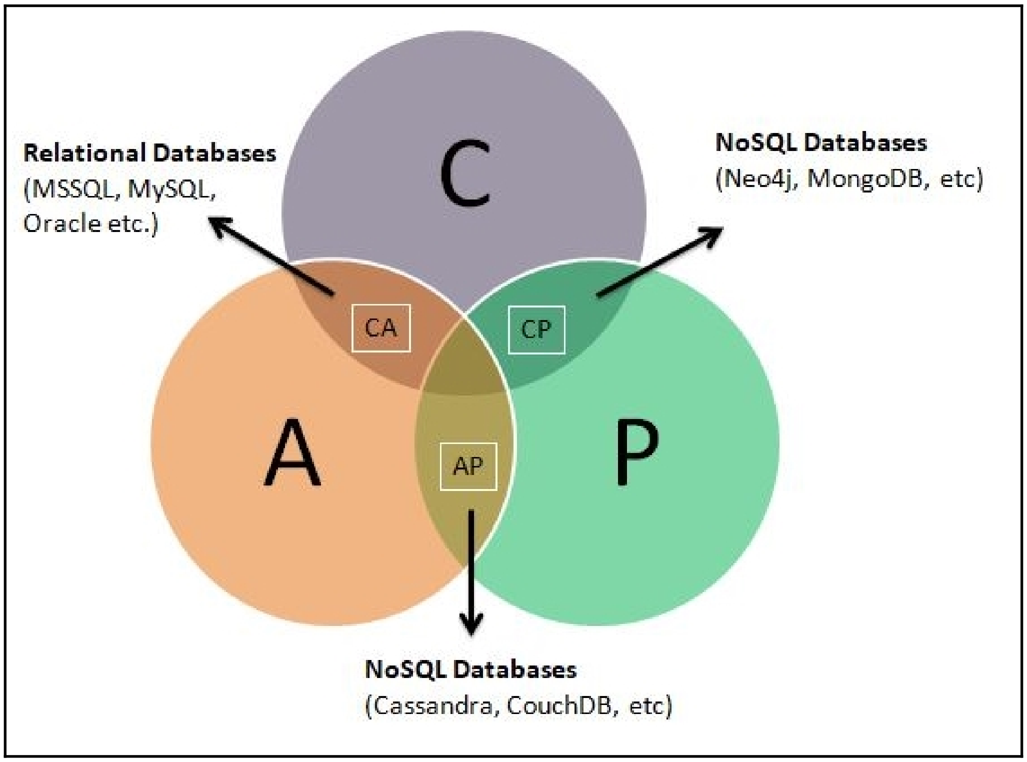
Now we will discuss the CAP theorem, which is related to NoSQL database, and ACID properties, which are related to SQL database, in order to understand some of the differences between NoSQL and Relational databases, as mentioned earlier.

The CAP theorem

CAP stands for **C**onsistency, **A**vailability, and **P**artition. It is also known as **Brewer's Theorem**, after the computer scientist Eric Brewer. The theory states that a distributed data system should ideally provide these three guarantees—consistency, availability, and partition. But in the real world, it is not possible to achieve all three of these. The following is an explanation of these three guarantees:

- **Consistency**: The data source should provide the most recent data after every successful write. It should make data updates across the whole cluster to provide up to date changes while accessing any data copy.
- **Availability**: It means that there is no downtime for the system and the service is always available to serve any request.
- **Partition tolerance**: It means the service will continue to serve requests even if any of the server partitions of a certain group become unavailable because of any reason. It means that server nodes will not be able to talk to each other, but they may still be able to service the client request. The issue may occur because of hardware failure, network failure, or delay in delivering network packets.

For NoSQL or any distributed data source, it is impossible to provide all three guarantees. That is why different NoSQL database frameworks provide different combinations of the CAP theorem. Some provide **consistency with availability** (**CA**), some provide **consistency with partitioning** (**CP**) and some provide **availability with partitioning** (**AP**). The following diagram illustrates these combinations:



Explanations of these combinations are as follows:

- **CA**: This configuration guarantees the consistency of data. It make sure that all the server nodes are available and are able to talk to each other, otherwise the service itself becomes unable to service any requests until the all the desired nodes are up and running.
- **CP**: In this configuration, even though it will have sites in different geographical locations, it will make sure that the data available is consistent across the cluster.
- **AP**: In this configuration, it will make sure that the service is always available with proper partitioning. It will not guarantee that the data returned by the service is consistent.

Examples of CA are Relational Databases such as MySQL, MSSQL, Oracle, and so on. For CP, examples are Neo4j and MongoDB, and examples of AP are Cassandra, CouchDB, and so on.

Let's now explore what a transaction is with respect to a database, and what the ACID properties of a transaction are.

The ACID properties

ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability. These are the properties that a transaction should have to reach the mark of a fully reliable operation. Maybe you are now thinking what actually is a transaction? It is a small bunch of code or a mini program that should run all together to keep data consistent on an existing database. A transaction that has the ACID properties will ensure the integrity of data during all of its internal operations. The following is an explanation of the individual properties:

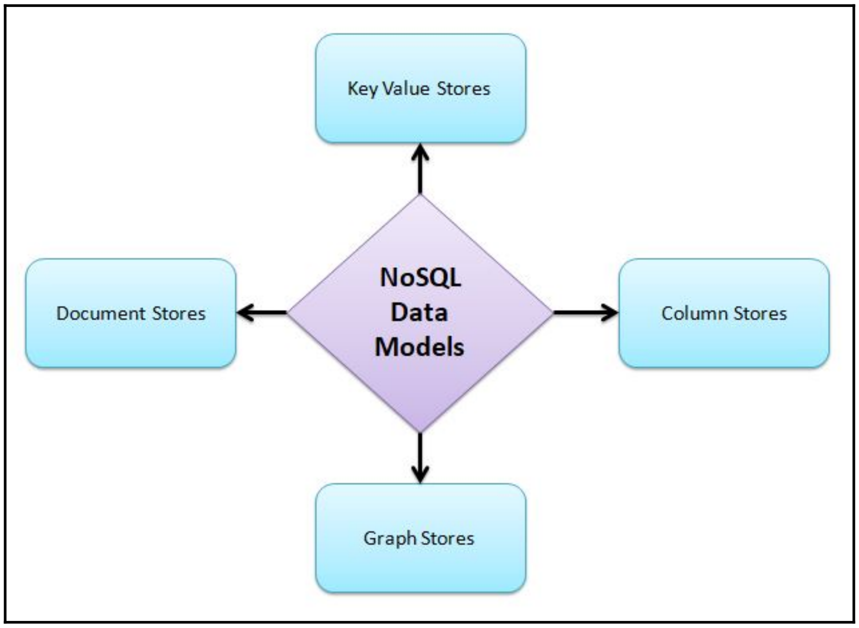
- **Atomicity**: It means if a transaction contains multiple reads or writes, all of them should take place. Otherwise, if any step generates an error or fails due to any reason, all the previous steps should roll back to their original state. In other words, a transaction should either execute fully or not execute at all.
- **Consistency**: A transaction should keep the integrity of data and leave the system in a valid state at the end of a transaction, so that if it completes successfully, the data will be consistent, and if there is any error, the system will return to its last state, which was already a valid state.
- **Isolation**: It means that at any given time only one transaction should take place on the required set of data—otherwise, suppose that two writes are happening at any given time; the data may lose its meaning.
- **Durability**: It means that any changes that happen as a result of a successfully transaction are permanent, no matter what happens to the server later on. All the steps done during a transaction should be logged so that, in the case of any failure, those steps can be recreated to restore their changes.

Let's now try to understand ACID properties using a banking transaction example. Let's suppose the account ABC wants to transfer $150 to the account XYZ. Currently, the available balance in account ABC is $1000. This transaction contains two steps; the first one is to debit the amount from the ABC account and the second one is to credit it to the XYZ account. The first steps executed successfully and debited the ABC account, and now the available balance is $850. But the second step fails and the available balance will remain the same for the XYZ account. This contains a serious risk for the bank and its reputation. To overcome this issue, a transaction that has ACID properties will rollback the debit and the amount in the ABC account will be back to $1000.

All the relational databases ensure that their transactions must have the ACID properties, whereas most of the NoSQL databases only support part of these properties.

Data models in NoSQL

Data model means the way that any framework internally stores and processes data. Every NoSQL database framework follows a data model to manipulate unstructured data. As of now, there are more than 150 frameworks available on the market. If we classify them, they will be broadly covered under four models, as illustrated in the following diagram and discussed as follows:



Key-value data stores

NoSQL databases built on this type of data model store data in the form of key-value pairs, as the name states. The key becomes the unique ID to insert and retrieve data whereas the value contains the actual data. This data can be of any form, such as JSON, video, audio, log file, and so on, making it flexible for storing any type of data. Its simplistic design increases its performance while reading and writing data. This type of NoSQL database is normally built for high availability and data partitioning, which is an AP configuration as per the CAP theorem.

It doesn't support transactions, which means that the data may not be reliable and consistent. These databases are not suitable when consistency and reliability is seriously required, such as in the banking application, or any type of payment solution. Now it's up to the solution architect to decide whether the data consistency and reliability is more important than replication and availability in the application.

Examples of key-value data store NoSQL Databases are Redis, DynamoDB, Riak, Azure Table Storage.

Document store

This type of databases is mainly designed to store semi structured data, such as JSON, XML, BSON, and so on. It is similar to key-value data stores as it generates a unique key for each row to associate it with semi structured document data. They are widely used in content management systems, mobile applications.

Examples of document store NoSQL Databases are MongoDB, Elasticsearch, ArangoDB, Azure Document DB, and so on.

Column stores

Column stores are also commonly known as **wide column stores**. They stores data in tabular form. They organize data in column form instead of row form, similar to relational databases, but the type and name of the columns may vary from row to row. It is sort of an inverse of the traditional systems. It is a highly scalable and very high performance architecture.

Examples of column store NoSQL databases are Cassandra, HBase, Hypertable, Amazon SimpleDB, Flink, and so on.

Graph stores

In this data model, each record is treated as a node and all nodes are linked together to form a relationship. It will help provide a more sophisticated and presentable form of data. It will evolve and help to identify different patterns as the data increases over time.

Examples of Graph Store NoSQL Databases are Neo4j, GraphBase, InfoGrid, and so on.

We have now learned enough theory about NoSQL databases. Let's now start with some practical stuff. Following are the commonly used NoSQL database frameworks of each data model type; we will go through their basic set up, commands, and data manipulation queries.

Apache Cassandra

Cassandra is one of the most widely used NoSQL databases from column data stores. We have already discussed the different functionalities and benefits of Cassandra in the previous chapter. Let's now do some hands-on.

Installation

First of all, we will go through a step-by-step guide for installing Cassandra, as follows:

1. Download the Cassandra package from the website `https://cassandra.apache.org/download/`. The Current stable release is 3.11.1.
2. Extract the package by executing the following command. It will be extracted in the directory `apache-cassandra-3.11.1`. In our case, the full path will be `/home/hadoopadmin/apache-cassandra-3.11.1/`. The following command illustrates this process:

   ```
   $ tar zxvf apache-cas sandra-3.11.1-bin.tar.gz
   ```

3. Set up the environment variable by executing the following command:

   ```
   $ gedit ~/.bashrc
   ```

4. Copy the following line at the end of the file and save it. Make sure to change the path as per your installation:

   ```
   # Cassandra Environment
   export CASSANDRA_HOME="/home/hadoopadmin/apache-cassandra-3.11.1/"
   export PATH=$PATH:$CASSANDRA_HOME/bin
   ```

5. Reload the environment file by executing the following command for your changes to take effect. Alternatively, you can restart your computer:

   ```
   $ source ~/.bashrc
   ```

TIP

You can edit the default configuration of Cassandra from the `cassandra.yaml` file. It exists inside the `cassandra/conf` folder. There are many configuration properties you can modify, such as cluster name, location of data directory, log directory, and some tuning information, such as heap, memory, and so on.

Starting Cassandra

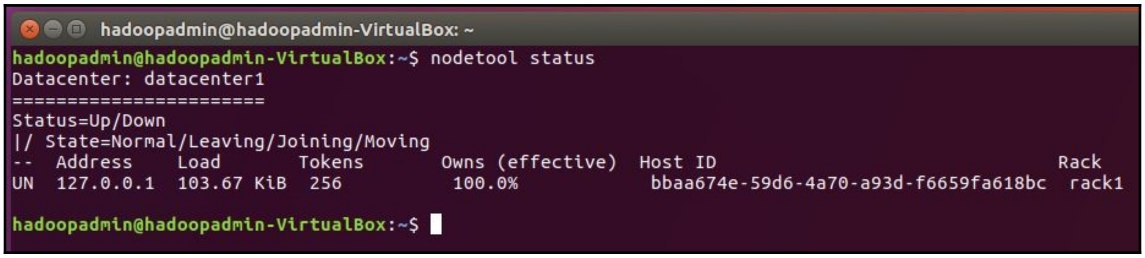
Now, we have done the basic setup of Apache Cassandra. It's time to start the service and test whether everything is working properly or not. In order to start Cassandra, execute the following command:

```
$ cassandra
```

It will display some logging information in the terminal. Once it returns to the prompt, it means that Cassandra has been started. To verify that it is up and running, execute the following command:

```
$ nodetool status
```

You will see an output similar to the one given in the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID                               Rack
UN  127.0.0.1  103.67 KiB  256          100.0%            bbaa674e-59d6-4a70-a93d-f6659fa618bc  rack1

hadoopadmin@hadoopadmin-VirtualBox:~$
```

TIP

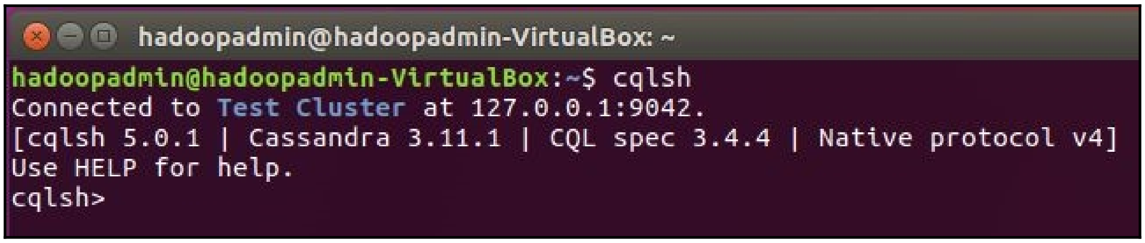
You can see the status `UN`, which means it is *up* and in the *normal* state against the node `127.0.0.1`.

The Cassandra Query Language – CQL

At this stage, we have successfully started our Cassandra server. It is time to start playing with it and get familiar with the environment. The built in tool to work on Cassandra is the **Cassandra Query Language** (**CQL**). It is similar to SQL but is based on the Cassandra's predefined functions. To start the CQL terminal, execute the following command:

```
$ cqlsh
```

Once you execute the previous command, you will see a welcome message that will show you the Cassandra version and CQL version as well as the cluster that you are connected to, as shown in the following screenshot:

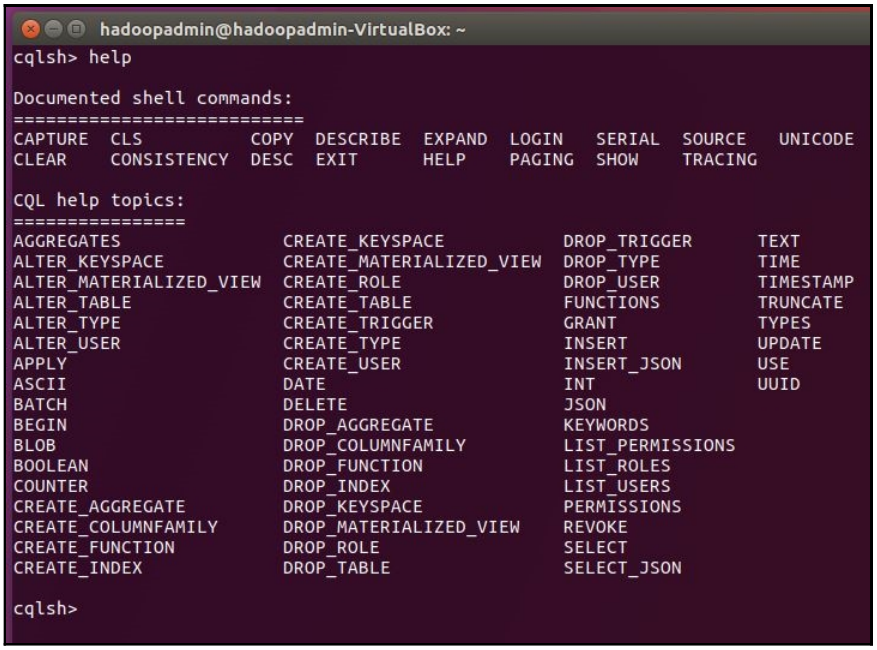


```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.1 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

From now on, whatever commands we execute, it will be done on this `cqlsh` prompt.

The help command

The first command that we are going to learn is the `help` command. It is the most useful command to get you out of any trouble. You don't need to learn each and every command syntax.; just use it to find out the options and different parameters related to any command. Execute the `help` command to list the output shown in the following screenshot:

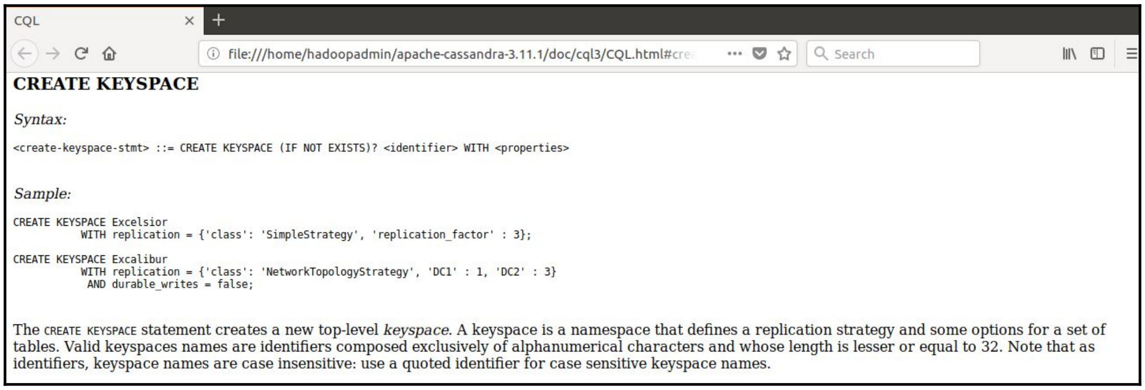


```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh> help

Documented shell commands:
===========================
CAPTURE  CLS          COPY  DESCRIBE  EXPAND  LOGIN   SERIAL  SOURCE   UNICODE
CLEAR    CONSISTENCY  DESC  EXIT      HELP    PAGING  SHOW    TRACING

CQL help topics:
================
AGGREGATES               CREATE_KEYSPACE           DROP_TRIGGER      TEXT
ALTER_KEYSPACE           CREATE_MATERIALIZED_VIEW  DROP_TYPE         TIME
ALTER_MATERIALIZED_VIEW  CREATE_ROLE               DROP_USER         TIMESTAMP
ALTER_TABLE              CREATE_TABLE              FUNCTIONS         TRUNCATE
ALTER_TYPE               CREATE_TRIGGER            GRANT             TYPES
ALTER_USER               CREATE_TYPE               INSERT            UPDATE
APPLY                    CREATE_USER               INSERT_JSON       USE
ASCII                    DATE                      INT               UUID
BATCH                    DELETE                    JSON
BEGIN                    DROP_AGGREGATE            KEYWORDS
BLOB                     DROP_COLUMNFAMILY         LIST_PERMISSIONS
BOOLEAN                  DROP_FUNCTION             LIST_ROLES
COUNTER                  DROP_INDEX                LIST_USERS
CREATE_AGGREGATE         DROP_KEYSPACE             PERMISSIONS
CREATE_COLUMNFAMILY      DROP_MATERIALIZED_VIEW    REVOKE
CREATE_FUNCTION          DROP_ROLE                 SELECT
CREATE_INDEX             DROP_TABLE                SELECT_JSON

cqlsh>
```

It will list down all the basic commands that `cql` supports. It will also list down the help topics that you can explore directly from command. For example, if you want to look for how to create a keyspace topic, execute the following command. It will open the default browser window with a sample and explanation:

```
cqlsh> help CREATE_KEYSPACE
```

The following screenshot shows the help topic that we just tried to explore:


CQL
file:///home/hadoopadmin/apache-cassandra-3.11.1/doc/cql3/CQL.html#cre
Search

CREATE KEYSPACE

Syntax:

```
<create-keyspace-stmt> ::= CREATE KEYSPACE (IF NOT EXISTS)? <identifier> WITH <properties>
```

Sample:

```
CREATE KEYSPACE Excelsior
          WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};

CREATE KEYSPACE Excalibur
          WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1' : 1, 'DC2' : 3}
           AND durable_writes = false;
```

The CREATE KEYSPACE statement creates a new top-level *keyspace*. A keyspace is a namespace that defines a replication strategy and some options for a set of tables. Valid keyspaces names are identifiers composed exclusively of alphanumerical characters and whose length is lesser or equal to 32. Note that as identifiers, keyspace names are case insensitive: use a quoted identifier for case sensitive keyspace names.

Basic commands

Sometimes, getting a basic detail out of something may be very painful. The following are some commands that will be helpful in navigating through Cassandra and extracting the information:

- **Describe**: The purpose of this command is to provide the relevant information related to the postfix object. The following are the multiple flavors of this command, which display information related to a specific object:

```
// It will list the basic information related to your cluster.
cqlsh> describe Cluster

// It will display the keyspaces define in this Cassandra Cluster
cqlsh> describe Keyspaces

// It will display all the tables created within a specific
keyspace
cqlsh> describe Tables

// It will describe definition of a specific table. What are its
column names,
// what are their types, and so on
cqlsh> describe Table tablename

// It will display all the user defined types in Cassandra.
cqlsh> describe Types

// It will display a definition of a specific typename.
```

```
cqlsh> describe Type typename
```

- **Exit**: This command is used to exit from the `cqlsh` terminal. The following is the syntax of `exit` command:

```
cqlsh> exit
```

- **Show**: Similar to `describe` command, it is also used to display different information but mostly related to cluster itself:

```
// It will show the current Cassandra version and cql version
information
cqlsh> show version

// It will show the current session host information
cqlsh> show host
```

- **Expand**: It is used to display information in rows instead of columns where required:

```
// To enable expand feature in current terminal
cqlsh> expand on

// Toggle expand interface off
cqlsh> expand off
```

- **Source**: This command is used to read lists of command from a text file and execute them directly in `cqlsh`:

```
// You can replace the file path with your own file
cqlsh> source '/home/user/command_list'
```

Data manipulation

Now that the Cassandra environment is ready, it's time to play around with some basic create, alter, and delete operations against database objects and data itself.

Creating, altering, and deleting a keyspace

First of all, we will learn how to create a keyspace. The **keyspace** is similar to a database in our traditional system. Execute the following command to create a `keyspace`:

```
cqlsh> Create Keyspace testKeyspace With replication =
{'class':'SimpleStrategy', 'replication_factor':3};
```

This command will create a keyspace named `testKeyspace`. The main point to note here is that we have selected the `SimpleStrategy` class with replication factor set to `3`. It means that every piece of information you store in Cassandra will be replicated in three different node servers within the same data center. If you have multiple data centers and you want to define different replication factor for each data center, you can use the `NetworkTopologyStrategy` class.

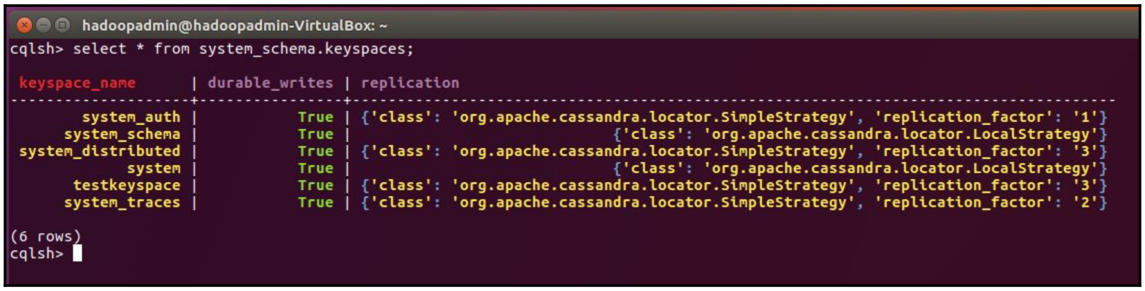
To test whether the `keyspace` has been created successfully, execute the following command:

```
cqlsh> describe keyspaces
```

It will list down all the available keyspace in the cluster; `testKeyspace` must be part of this list. If want to list down the full details against each `keyspace`, what its replication factor is, and which class you selected while creating your `keyspaces`, execute the following command:

```
cqlsh> select * from system_schema.keyspaces;
```

It will give you an output similar to the following screenshot:



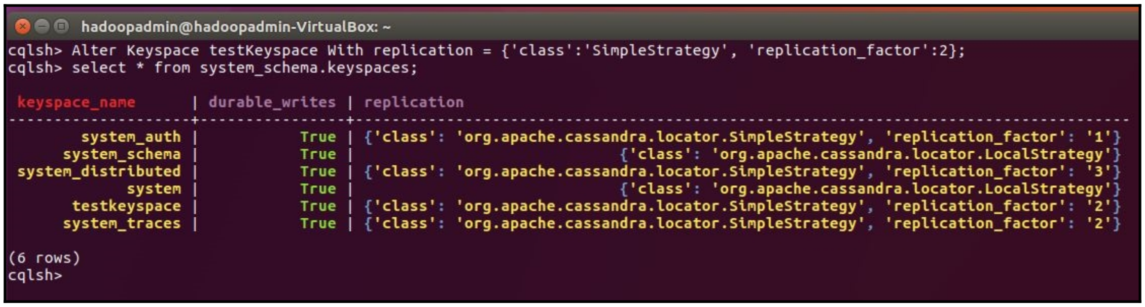
```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh> select * from system_schema.keyspaces;

 keyspace_name      | durable_writes | replication
--------------------+----------------+-------------------------------------------------------------------------------------
        system_auth |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
      system_schema |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
 system_distributed |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
             system |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
      testkeyspace |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
      system_traces |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}

(6 rows)
cqlsh>
```

Conventionally, any keyspace starting with the `system` keywords belongs to Cassandra's internal operations and is better left untouched. If modified directly, it may lead to an unstable installation.

Let's suppose that now we want to change the replication factor for our newly created `keyspace` from `3` to `2`. It may be because of any reason. To do this alteration, execute the following command. It is very similar to the `create` command:

```
cqlsh> Alter Keyspace testKeyspace With replication =
{'class':'SimpleStrategy', 'replication_factor':2};
```

The following screenshot displays the output of this alteration and the `verification` command:

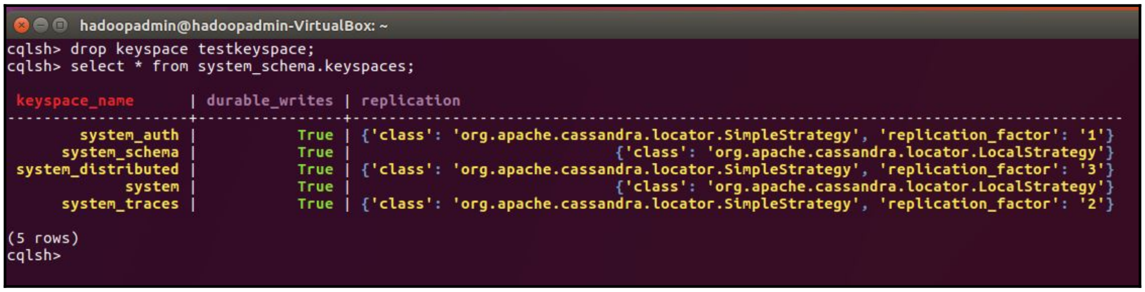


```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh> Alter Keyspace testKeyspace With replication = {'class':'SimpleStrategy', 'replication_factor':2};
cqlsh> select * from system_schema.keyspaces;

 keyspace_name      | durable_writes | replication
--------------------+----------------+-------------------------------------------------------------------------------------
        system_auth |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
      system_schema |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
 system_distributed |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
             system |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
       testkeyspace |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}
      system_traces |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}

(6 rows)
cqlsh>
```

Let's now suppose that you no longer require this newly created `keyspace`. In order to delete the `keyspace`, execute the following command:

```
cqlsh> drop keyspace testkeyspace;
```

The following screenshot displays the output of `drop keyspace` and the `verification` command:



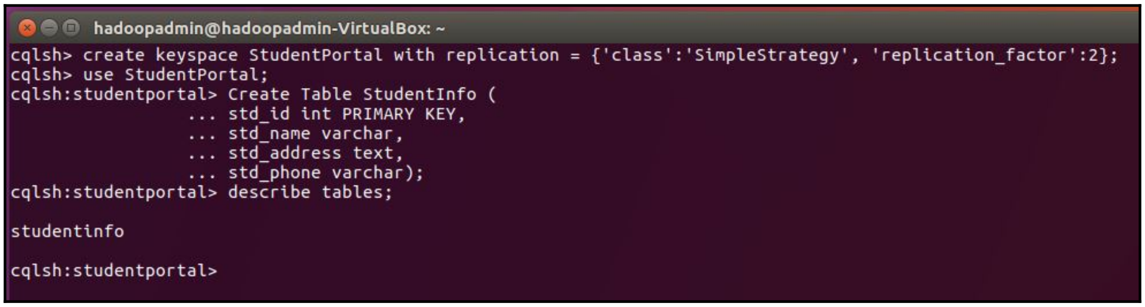
```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh> drop keyspace testkeyspace;
cqlsh> select * from system_schema.keyspaces;

 keyspace_name      | durable_writes | replication
--------------------+----------------+-------------------------------------------------------------------------------------
        system_auth |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
      system_schema |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
 system_distributed |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
             system |           True |                             {'class': 'org.apache.cassandra.locator.LocalStrategy'}
      system_traces |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}

(5 rows)
cqlsh>
```

Creating, altering, and deleting tables

Let's now create a new keyspace, `StudentPortal`, using the previously discussed command. It's now time to create our first table. This table contains some basic information related to a student, for example, student ID, student name, address, phone number, and so on. Execute the following command to create the student table:

```
cqlsh> Use StudentPortal;
cqlsh:studentportal> Create table StudentInfo (std_id int PRIMARY KEY,
std_name varchar, std_address text, std_phone varchar);
```

The following screenshot displays the output of the previously executed command as well as the `verification` command:

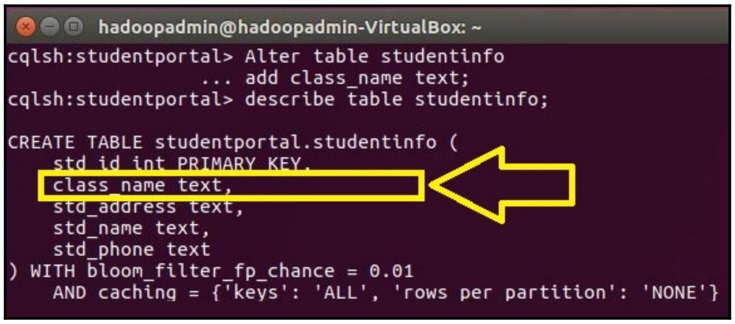


```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh> create keyspace StudentPortal with replication = {'class':'SimpleStrategy', 'replication_factor':2};
cqlsh> use StudentPortal;
cqlsh:studentportal> Create Table StudentInfo (
                 ... std_id int PRIMARY KEY,
                 ... std_name varchar,
                 ... std_address text,
                 ... std_phone varchar);
cqlsh:studentportal> describe tables;

studentinfo

cqlsh:studentportal>
```

Let's assume that you are required to add the new column `ClassName` to the `studentinfo` table. For this purpose, execute the following command:

```
cqlsh:studentportal> Alter table studentinfo add class_name text;
```

The following screenshot displays the output of the previously executed command and the `verification` command:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh:studentportal> Alter table studentinfo
                 ... add class_name text;
cqlsh:studentportal> describe table studentinfo;

CREATE TABLE studentportal.studentinfo (
    std_id int PRIMARY KEY,
    class_name text,
    std_address text,
    std_name text,
    std_phone text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
```

In cases where you need to remove a column from the table, execute the following command:

```
cqlsh:studentportal> alter table studentinfo drop class_name;
```

Assume that you no longer need this table and its data. Now it's time to delete it from our database. For this purpose, execute the following command:

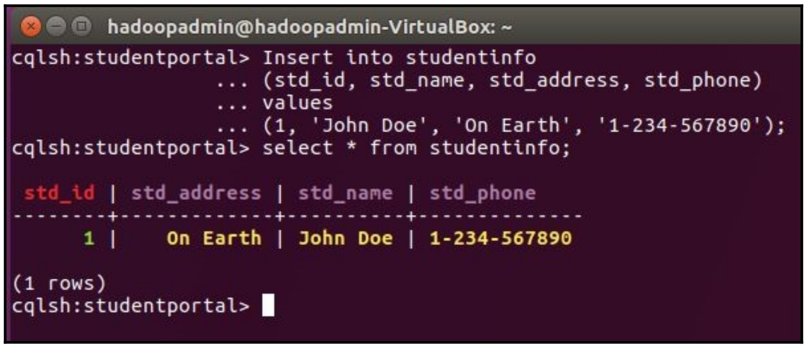
```
cqlsh:studentportal> drop table studentinfo;
```

Inserting, updating, and deleting data

Up to this stage, we are able to create keyspaces and tables, and have the information about how to alter and delete them, if needed. Now it's time to start inserting data into our database. Before executing the following command, make sure you have recreated the `studentinfo` table, so that you will not encounter any errors. To insert our first record, execute the following command:

```
cqlsh:studentportal> Insert into studentinfo (std_id, std_name,
std_address, std_phone) values (1, 'John Doe', 'On Earth', '1-234-567890')
```

The following screenshot displays the output of the previous query as well as the `verification` command:

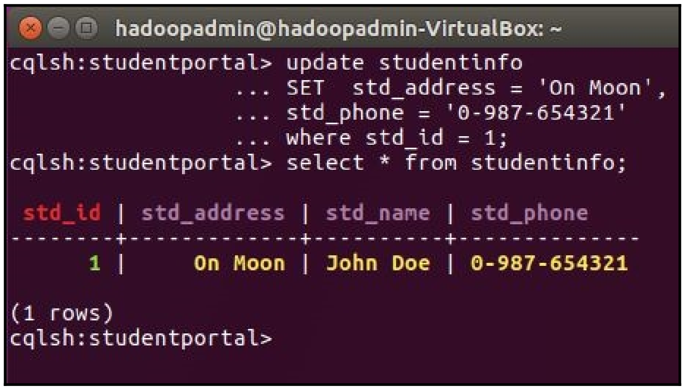


```
cqlsh:studentportal> Insert into studentinfo
                 ... (std_id, std_name, std_address, std_phone)
                 ... values
                 ... (1, 'John Doe', 'On Earth', '1-234-567890');
cqlsh:studentportal> select * from studentinfo;

 std_id | std_address | std_name | std_phone
--------+-------------+----------+--------------
      1 |    On Earth | John Doe | 1-234-567890

(1 rows)
cqlsh:studentportal>
```

Let's assume that the student has relocated to some other location and wants to update the address and the phone number. In order to perform such action, execute the following query:

```
cqlsh:studentportal> update studentinfo SET  std_address = 'On Moon',
std_phone = '0-987-654321' where std_id = 1;
```

The following screenshot displays the output of the previous query as well as the `verification` command:

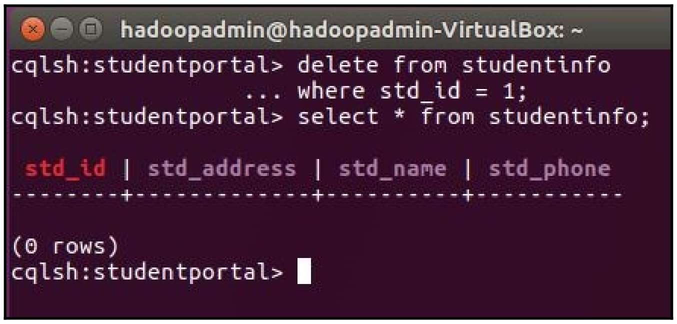


```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh:studentportal> update studentinfo
                 ... SET  std_address = 'On Moon',
                 ... std_phone = '0-987-654321'
                 ... where std_id = 1;
cqlsh:studentportal> select * from studentinfo;

 std_id | std_address | std_name | std_phone
--------+-------------+----------+--------------
      1 |     On Moon | John Doe | 0-987-654321

(1 rows)
cqlsh:studentportal>
```

Now consider a case where a student has left the school, and you no longer want to keep data related to that student. To `delete` the record, execute the following query:

```
cqlsh:studentportal> delete from studentinfo where std_id = 1
```

The following screenshot displays the output of the previous query as well as the `verification` command:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
cqlsh:studentportal> delete from studentinfo
                 ... where std_id = 1;
cqlsh:studentportal> select * from studentinfo;

 std_id | std_address | std_name | std_phone
--------+-------------+----------+-----------

(0 rows)
cqlsh:studentportal> 
```

You have now got practical experience in setting up Cassandra, as well as hands-on experience with many CQL commands and queries. This tutorial has covered all the major scenarios, including setting up the environment, handling different database objects, and data manipulation.

In the next topic, we will learn about MongoDB, which also belongs to NoSQL world, but has a different data model.

The MongoDB database

MongoDB is a NoSQL database based on the document store data model. It stores data in a JSON style formatted document. In MongoDB everything is either collection or document. In order to understand MongoDB terminology, **collection** is an alternate word for table, whereas **document** is an alternate word for rows.

Installing MongoDB

We are now going to install MongoDB's community edition. Follow the step-by-step guide to successfully install it on your server:

1. Ubuntu package management require that distributors sign packages with GPG keys for consistency and authenticity within packages. Execute the following command to import the MongoDB GPG key:

   ```
   $ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
   0C49F3730359A14518585931BC711F9BA15703C6
   ```

2. Execute the following command to create a MongoDB source list:

   ```
   $ echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu
   xenial/mongodb-org/3.4 multiverse" | sudo tee
   /etc/apt/sources.list.d/mongodb-org-3.4.list
   ```

3. Execute the following command to `update` the repository package details, as we have now added MongoDB repository to the list:

   ```
   $ sudo apt-get update
   ```

4. Now it's time to install MongoDB by issuing the following command. It will automatically install the latest stable version of MongoDB:

   ```
   $ sudo apt-get install -y mongodb-org
   ```

We have now successfully installed MongoDB on our server. We will now start the service and test whether it is working properly or not.

Starting MongoDB

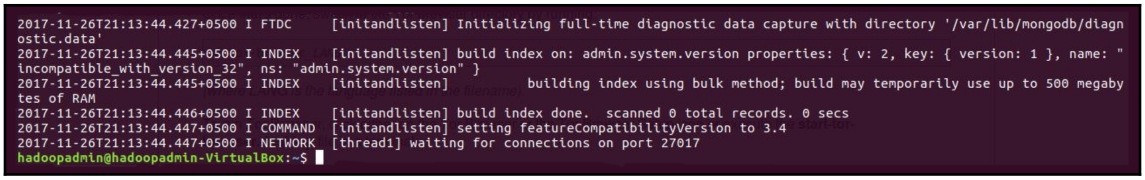
It's time to test whether MongoDB installed successfully. In order to do so, issue the following command:

```
$ sudo service mongod start
```

If this returns to the terminal without any error, it means that MongoDB has started successfully. To double check, MongoDB will have created a log file in `/var/log/mongodb/mongod.log`. Open it with any test editor, and if it shows a message that it's waiting for a connection, this means it is working properly. Alternatively, you can issue a `cat` command to list the contents of the file as follows:

```
$ cat  /var/log/mongodb/mongod.log
```

The following screenshot illustrates the output of the previous command:



```
2017-11-26T21:13:44.427+0500 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagn
ostic.data'
2017-11-26T21:13:44.445+0500 I INDEX    [initandlisten] build index on: admin.system.version properties: { v: 2, key: { version: 1 }, name: "
incompatible_with_version_32", ns: "admin.system.version" }
2017-11-26T21:13:44.445+0500 I INDEX    [initandlisten]          building index using bulk method; build may temporarily use up to 500 megaby
tes of RAM
2017-11-26T21:13:44.446+0500 I INDEX    [initandlisten] build index done.  scanned 0 total records. 0 secs
2017-11-26T21:13:44.447+0500 I COMMAND  [initandlisten] setting featureCompatibilityVersion to 3.4
2017-11-26T21:13:44.447+0500 I NETWORK  [thread1] waiting for connections on port 27017
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Whenever you want to `stop` the server, issue the following command:

```
$ sudo service mongod stop
```

In order to `restart` the server, issue the following command:

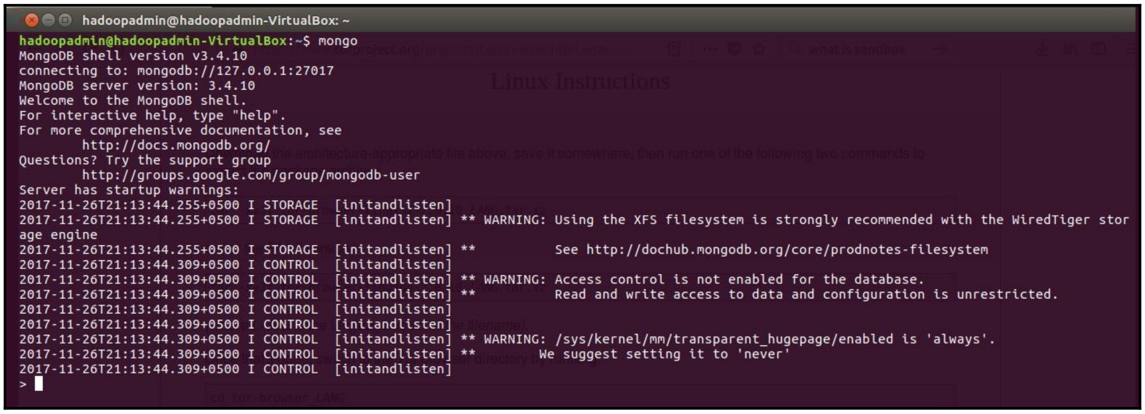
```
$ sudo service mongod restart
```

Working on MongoDB

Now that we have successfully installed and started our MongoDB server, it's time to enter into the world of MongoDB. Type in the following command to start a MongoDB shell session:

```
$ mongo
```

This command will start your session to interact with MongoDB. During startup, it will display its version, IP address, the port it is listening on, a welcome message, and some logging information. The following screenshot illustrates the output terminal:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ mongo
MongoDB shell version v3.4.10
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.10
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2017-11-26T21:13:44.255+0500 I STORAGE  [initandlisten]
2017-11-26T21:13:44.255+0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger stor
age engine
2017-11-26T21:13:44.255+0500 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten]
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten]
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten]
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-11-26T21:13:44.309+0500 I CONTROL  [initandlisten]
> 
```

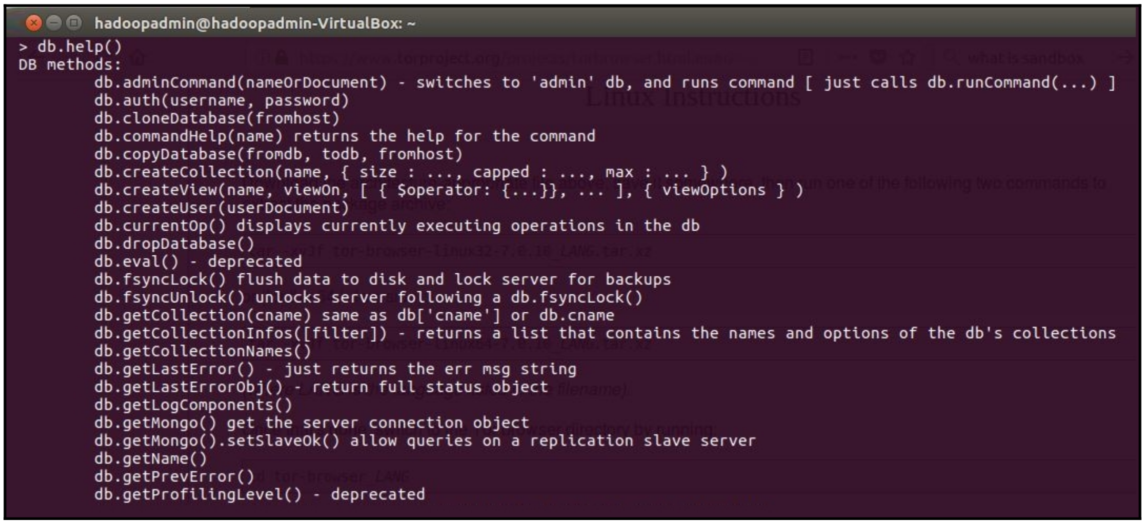
In MongoDB, whatever operation we perform in its shell is a function. Some functions are predefined as static and some are dynamic and change based on the collection you are working on.

The help command

While working on any application, the most important thing is that if you don't remember the commands to get the desired result, then you know how to get help. Of course, you can search over the internet, but a mark of quality in a good application is to have its own built in help system to keep you going. MongoDB has a default list of commands with a brief description, ready for you to display whenever required. To list them, execute the following command on the MongoDB shell session that we have just started:

```
> db.help()
```

The following screenshot illustrates the output of this command:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [ just calls db.runCommand(...) ]
        db.auth(username, password)
        db.cloneDatabase(fromhost)
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost)
        db.createCollection(name, { size : ..., capped : ..., max : ... } )
        db.createView(name, viewOn, [ { $operator: {...}}, ... ], { viewOptions } )
        db.createUser(userDocument)
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase()
        db.eval() - deprecated
        db.fsyncLock() flush data to disk and lock server for backups
        db.fsyncUnlock() unlocks server following a db.fsyncLock()
        db.getCollection(cname) same as db['cname'] or db.cname
        db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
        db.getCollectionNames()
        db.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSlaveOk() allow queries on a replication slave server
        db.getName()
        db.getPrevError()
        db.getProfilingLevel() - deprecated
```

Basic commands

Let's now learn some basic commands to make our way in MongoDB:

- To list down all the `databases` that exist in the current setup of MongoDB, execute either of the following commands:

```
> show databases;
> show dbs;
```

- To list down all the tables (collections) within a specific database, execute the following command:

```
// first switch to a database to explore its tables
> use databasename;
// To list down all the collection in a database
> show collections;
```

- To check in which database currently you are working, execute the following command:

```
> db
```

- To clear the screen, execute the following command:

```
> cls
```

Data manipulation

MongoDB is a NoSQL database, therefore it doesn't rely very much on defined schemas and objects. We will discuss some data manipulation commands that you can use for this purpose in the coming topics.

Creating and deleting databases

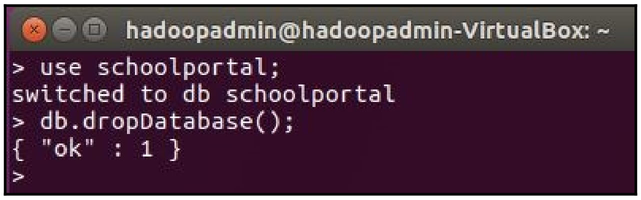
It is very easy to create a database. Just execute the `use` command to mention that from now on all commands will be executed in a specific database:

```
// The following command will check whether the database already exists.
// If not it will create a database with this name.
> use schoolportal;
```

If at any instance you want to delete a database, first you have switch to the database you want to delete, and then execute the following command:

```
// To switch to database that you want to delete
> use schoolportal;

// Now it's time to delete it
> db.dropDatabase();
```

The following screenshot illustrates the execution of the earlier command:



```
> use schoolportal;
switched to db schoolportal
> db.dropDatabase();
{ "ok" : 1 }
>
```

Creating and deleting collections

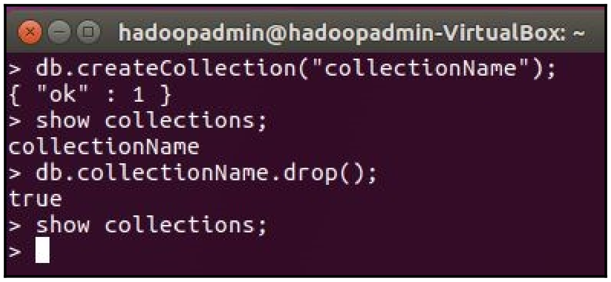
Collection is an another word for a table in MongoDB terminology. It is what holds the data. It doesn't require any schema while creating it. In order to create a collection, execute the following command:

```
> db.createCollection("collectionName");
```

In order to delete the collection and all the documents inside it, execute the following command:

```
// Replace the collectionName with the collection you want to remove
> db.collectionName.drop();
```

The following screenshot illustrates the output of the earlier commands with the `verification` command:



```
> db.createCollection("collectionName");
{ "ok" : 1 }
> show collections;
collectionName
> db.collectionName.drop();
true
> show collections;
>
```

The create, retrieve, update, delete operations

Every row is called a document in MongoDB terminology. You can have *n* number of documents in any collection. The following are the basic operations that you can perform in relation to any document:

- Each document can have a different number of columns. It's up to you how you place your data, under which collection and database. In order to insert a document, execute the following command:

  ```
  > db.employees.insertOne({"first_name":"John", "last_name":"Doe"});
  ```

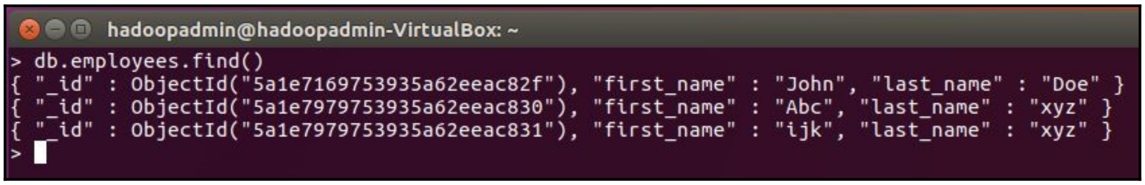
- To insert documents in bulk, you can use the function `insertMany` instead of `insertOne` as shown in the following snippet:

```
> db.employees.insertMany([
        {"first_name":"Abc", "last_name":"xyz"},
        {"first_name":"ijk", "last_name":"xyz"}
    ]);
```

- To retrieve all the records in the employees collection, execute the following command:

```
> db.employees.find()

// If you want to display results vertically, use the following
command
> db.employees.find().pretty()
```

The following screenshot displays the command in operation and the three records we have inserted just now:

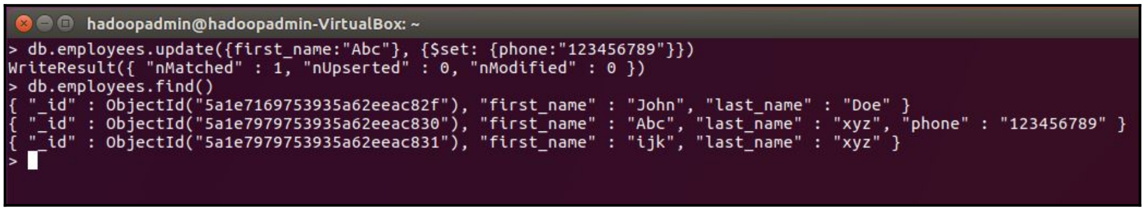


```
hadoopadmin@hadoopadmin-VirtualBox: ~
> db.employees.find()
{ "_id" : ObjectId("5a1e7169753935a62eeac82f"), "first_name" : "John", "last_name" : "Doe" }
{ "_id" : ObjectId("5a1e7979753935a62eeac830"), "first_name" : "Abc", "last_name" : "xyz" }
{ "_id" : ObjectId("5a1e7979753935a62eeac831"), "first_name" : "ijk", "last_name" : "xyz" }
>
```

- The beauty of MongoDB or NoSQL database is that there is no defined schema, so you can add columns on the fly and to only specific documents. Let's suppose you want to add a phone number for the employee `Abc`. For this purpose, issue the following command:

```
> db.employees.update({first_name:"Abc"}, {$set:
{phone:"123456789"}})

// Similarly to remove a column from a document,
// execute the following command.
> db.employees.update({first_name:"Abc"}, {$unset: {phone}})
```

The following screenshot displays the output as well as the `verification` command result:

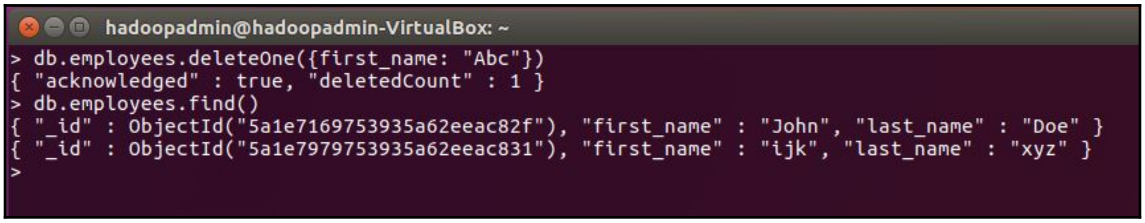


```
hadoopadmin@hadoopadmin-VirtualBox: ~
> db.employees.update({first_name:"Abc"}, {$set: {phone:"123456789"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.employees.find()
{ "_id" : ObjectId("5a1e7169753935a62eeac82f"), "first_name" : "John", "last_name" : "Doe" }
{ "_id" : ObjectId("5a1e7979753935a62eeac830"), "first_name" : "Abc", "last_name" : "xyz", "phone" : "123456789" }
{ "_id" : ObjectId("5a1e7979753935a62eeac831"), "first_name" : "ijk", "last_name" : "xyz" }
> 
```

- To delete any document from a collection, execute the following command. As a parameter, you can apply any filter. For the sake of example, we are going to delete a record where the first name is mentioned as `Abc`:

```
> db.employees.deleteOne({first_name: "Abc"})

// If your filter criteria return multiple documents,
// then execute the following command
> db.employees.deleteMany({first_name: "Abc"})
```

The following screenshot displays the output of the following command as well as the `verification` command:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
> db.employees.deleteOne({first_name: "Abc"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.employees.find()
{ "_id" : ObjectId("5a1e7169753935a62eeac82f"), "first_name" : "John", "last_name" : "Doe" }
{ "_id" : ObjectId("5a1e7979753935a62eeac831"), "first_name" : "ijk", "last_name" : "xyz" }
>
```

We have successfully gone through how to set up MongoDB and its major commands and functions, which will help you while dealing with MongoDB. In the next topic, we will go through the Neo4j database, which belongs to the graph data model in the NoSQL world.

Neo4j database

Neo4j belongs to the NoSQL database world and it's based on the graph store data model. It is very useful if you want to visualize your data. The graphical data visualization often leads to the identification of different patterns that are largely overlooked when we view data in tabular or textual formats. It is an open source software and the complete source code is available at GitHub.

Installing Neo4j

We will now install Neo4j. Follow this step-by-step guide to successfully install it in your server:

1. Execute the following to extract the GPG key, similar to what we did during the installation of MongoDB:

   ```
   $ wget -O - https://debian.neo4j.org/neotechnology.gpg.key | sudo
   apt-key add -
   ```

2. Execute the following command to create a Neo4j repository source list:

   ```
   $ echo 'deb http://debian.neo4j.org/repo stable/' | sudo tee -a
   /etc/apt/sources.list.d/neo4j.list
   ```

3. Execute the following command to update the repository package details, as we have now added the Neo4j repository to the list:

   ```
   $ sudo apt-get update
   ```

4. Now it's time to install Neo4j by issuing the following command. It will download almost 200 Mb of installation files:

   ```
   $ sudo apt-get install neo4j=3.3.0
   ```

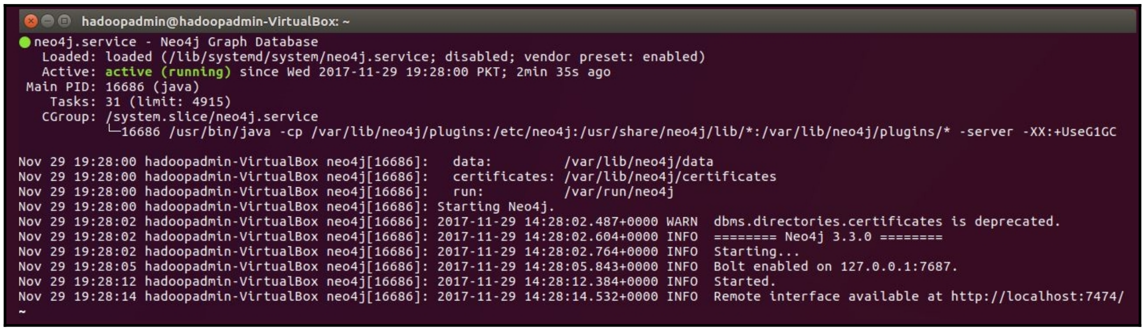
We have now successfully installed Neo4j on our server. We will now start the service and test it whether it is working properly.

Starting Neo4j

Once you finish the installation, it will automatically start Neo4j service in the backend. To verify whether it is active and working properly or not, issue the following command:

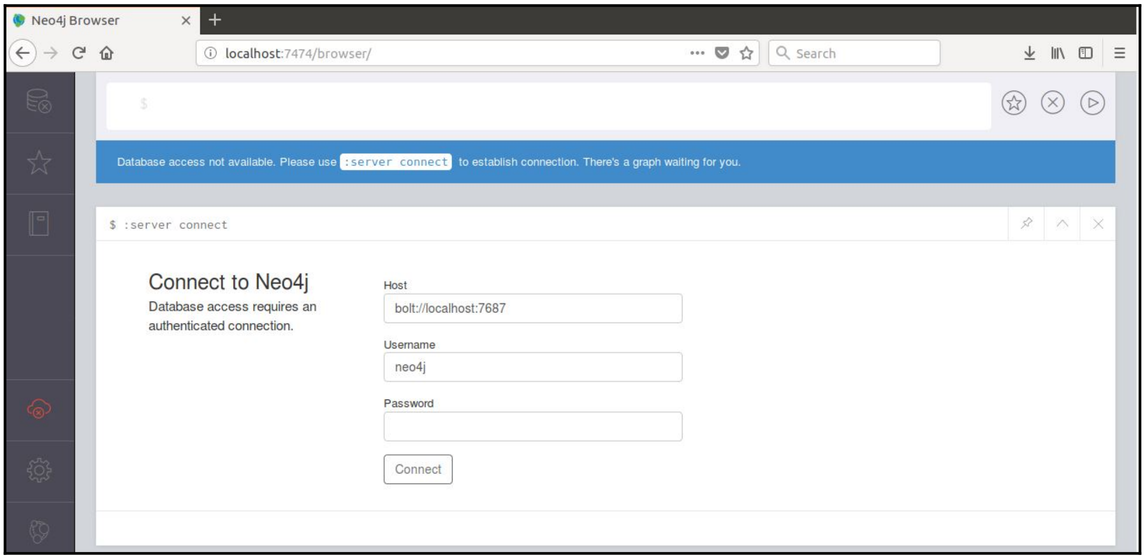
```
$ service neo4j status
```

The following screenshot displays the output of the earlier command. It shows that the service is active with some information messages:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
● neo4j.service - Neo4j Graph Database
   Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2017-11-29 19:28:00 PKT; 2min 35s ago
 Main PID: 16686 (java)
    Tasks: 31 (limit: 4915)
   CGroup: /system.slice/neo4j.service
           └─16686 /usr/bin/java -cp /var/lib/neo4j/plugins:/etc/neo4j:/usr/share/neo4j/lib/*:/var/lib/neo4j/plugins/* -server -XX:+UseG1GC

Nov 29 19:28:00 hadoopadmin-VirtualBox neo4j[16686]:   data:         /var/lib/neo4j/data
Nov 29 19:28:00 hadoopadmin-VirtualBox neo4j[16686]:   certificates: /var/lib/neo4j/certificates
Nov 29 19:28:00 hadoopadmin-VirtualBox neo4j[16686]:   run:          /var/run/neo4j
Nov 29 19:28:00 hadoopadmin-VirtualBox neo4j[16686]: Starting Neo4j.
Nov 29 19:28:02 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:02.487+0000 WARN  dbms.directories.certificates is deprecated.
Nov 29 19:28:02 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:02.604+0000 INFO  ======== Neo4j 3.3.0 ========
Nov 29 19:28:02 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:02.764+0000 INFO  Starting...
Nov 29 19:28:05 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:05.843+0000 INFO  Bolt enabled on 127.0.0.1:7687.
Nov 29 19:28:12 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:12.384+0000 INFO  Started.
Nov 29 19:28:14 hadoopadmin-VirtualBox neo4j[16686]: 2017-11-29 14:28:14.532+0000 INFO  Remote interface available at http://localhost:7474/
~
```

Neo4j has a browser based shell portal where you can perform many actions along with browsing the structure and running queries. To access this management portal, point your browser to the URL mentioned in the last line in the previous screenshot; `http://localhost:7474/`. It will display a login page as shown in the following screenshot:



Database access not available. Please use `:server connect` to establish connection. There's a graph waiting for you.

```
$ :server connect
```

Connect to Neo4j

Database access requires an authenticated connection.

Host: bolt://localhost:7687

Username: neo4j

Password:

The default username and password is `neo4j`. Once you log in for the first time, it will ask you to change it. Make note of the newly changed password as you will be needing it for future connectivity.

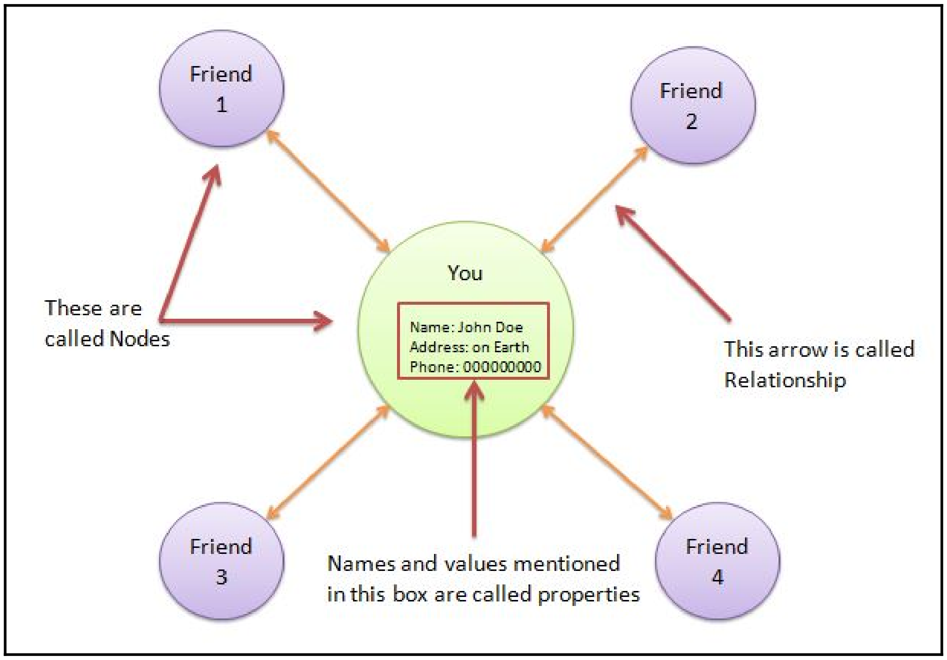
The cypher query language

The cypher query language is similar to SQL but specifically for generating graph outputs as results. It was developed by the *Neo Technology* company for Neo4j, but was then released as an open source project named **openCypher** so that other graph database can also utilize it in their projects. It translates very complicated traditional queries into simple CQL graph queries. So, let's get started.

There are three main components in a graph, namely:

- **Node**: This is the primary object in representing any data in graph form. You may consider it like a main element that makes to other objects. It represents entities such as person, event, place, thing, and so on. If you take an example of a social networking graph, node is a representation of you and your friends connected to each other.
- **Relationship**: It is the representation of how nodes are connected and what type of connection they are in. If we expand the earlier example, where you and your friends are connected to each other, relationship defines what type of connection it is—whether the connected one is a friend, colleague, business partner, and so on.
- **Properties**: Every node and relationship may have some properties. For example, your nodes in the earlier example may contain some information related to you, such as your name, address, phone number, and so on. They are in the form of key-value pairs.

The following diagram illustrates the earlier components in a graph:

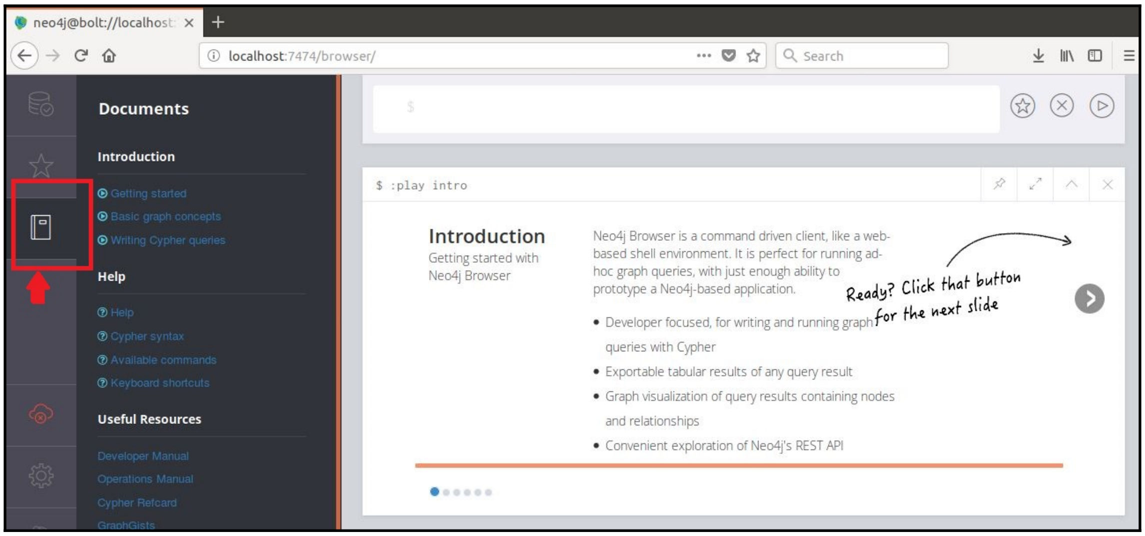


Now let's talk about the syntax of cypher. It comprises of three main clauses, namely:

- **MATCH**: It is where you build your graph that contain nodes and relationships. In other words, it contains the structure of the graph.
- **WHERE**: It contains the filter criteria that applies on the MATCH clause to output the desired data.
- **RETURN**: This contains the desired output of nodes, relationship, and properties in graphical form. If you apply any aggregation function in your return statement, then the output will be in tabular form.

Help

Neo4j's browser utility is built for documentation and sample data. It displays tutorials in a a nice step-by-step style to make you feel comfortable. To access this section, point your browser to the Neo4j browser shell and click on the book-like icon on the left, as illustrated in the following screenshot:



```
$ :play intro
```

Introduction
Getting started with Neo4j Browser

Neo4j Browser is a command driven client, like a web-based shell environment. It is perfect for running ad-hoc graph queries, with just enough ability to prototype a Neo4j-based application.

- Developer focused, for writing and running graph queries with Cypher
- Exportable tabular results of any query result
- Graph visualization of query results containing nodes and relationships
- Convenient exploration of Neo4j's REST API

Ready? Click that button for the next slide

Basic operations in Cypher

Let's now execute some Cypher queries to get hands-on experience. In the following examples, we will be creating, updating, and deleting nodes, relationships, and properties. In the following section, we will be dealing with example of social networking where we have many users and they may have the relationship of friend. Let's get started.

Creating nodes, relationships, and properties

In order to create a node, execute the following command:

```
// Single node creation with Title
$ Create (n:User)
```

Consider now that we have two users, `John` and `Andrew`. They become friends. In Neo4j, it's now time to create a relationship between them, so that when you query the database in a graph form, it will display a friend relationship between them. For this purpose, execute the following command:

```
// Search for the node type for which relationship is required
$ MATCH (a:User),(b:User)

// Filter the exact result
WHERE a.firstname = 'John' AND b.firstname = 'Andrew'

// Create a relationship titled Friend
CREATE (a)-[r:Friend]->(b)
RETURN r
```

To create properties, execute the following command:

```
// Single node creation with title as well as with properties
$ Create (n:User {firstname: 'John', lastname: 'Doe'} )
```

Updating nodes, relationships, and properties

As Neo4j is also a NoSQL database, it supports adding column or properties on the fly. Now let's assume that you want to add an `age` property to the existing node of the user `John`. To perform this action, execute the following command:

```
$ Merge (u:User {firstname:'John'})
set u.age = 30
Return u
```

This `Merge` command will first check whether this age property already exists or not. If not, it will create it for you. Otherwise, it will just update it with the value you mentioned in the query.

Deleting nodes, relationships, and properties

To delete a node, execute the following command. Make sure that there is no relationship defined for the node you are going to delete. If any relationships exist, the query will not execute and will throw an error:

```
// to delete all user nodes
$ Match (u:User)
Delete u

// to delete based on filter criteria
$ Match (u:User)
where u.name = 'John'
Delete u
```

To delete a node with a relationship, execute the following command:

```
$ Match (u:User)
where u.name = 'John'
Detach Delete u
```

To delete properties, execute the following command:

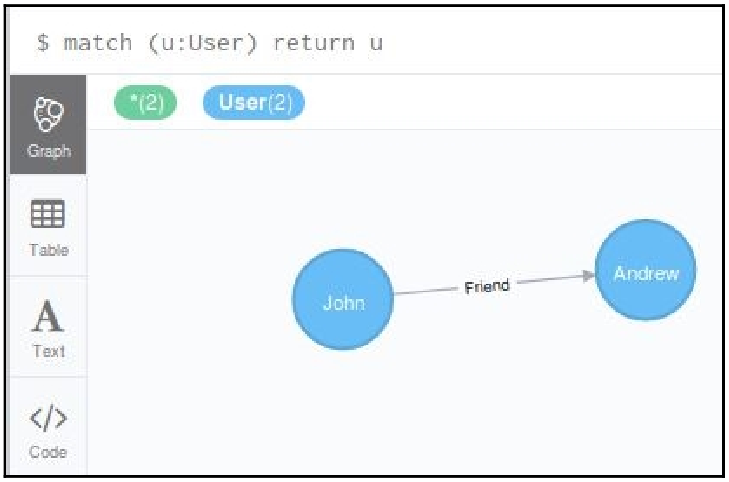
```
$ Match (u:User)
where u.firstname = 'John'
Remove u.lastname
Return u
```

Reading nodes, relationships, and properties

Neo4j supports and displays the result in both graph as well as table formats based on your queries and requirements. Let's start will some basic examples. Let's create a graph to list all friends of the user `John`. Execute the following query to retrieve the data in graph form:

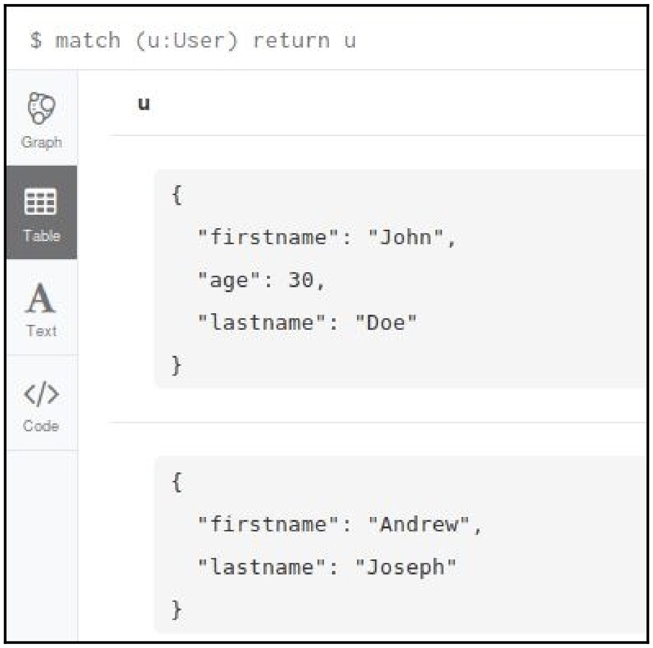
```
$ MATCH (u:User)
WHERE u.firstname = 'John'
RETURN u
```

The following screenshot illustrates the output of this query:



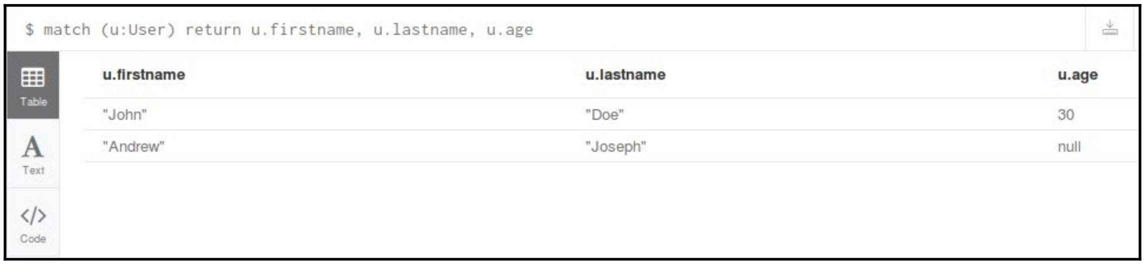
```
$ match (u:User) return u
```

You can also select the format of the result from the left navigation panel. If you select **Table**, it will give you the result in a nicely formatted tabular form, where each row contains all the properties belonging to that node. The following screenshot illustrate its output:



```
$ match (u:User) return u

u

{
  "firstname": "John",
  "age": 30,
  "lastname": "Doe"
}

{
  "firstname": "Andrew",
  "lastname": "Joseph"
}
```

Let's assume that you don't need the output in a graph form but you need some properties displayed as an output, similar to RDBMS results. For this purpose, in the return statement, instead of returning the full node, you can specify the properties you need. Neo4j will then return the result in a table format, as shown in the following screenshot:



```
$ match (u:User) return u.firstname, u.lastname, u.age
```

u.firstname	u.lastname	u.age
"John"	"Doe"	30
"Andrew"	"Joseph"	null

These are just basic examples to get you started with Neo4j. You can create more complex queries and graphs as you move forward.

Summary

In this chapter, we first learned about the NoSQL database and its advantages. We then went through a point by point comparison of NoSQL databases and RDBMS. We then discussed the CAP theorem and ACID properties of transactions and their relation to NoSQL database. We then discussed different data models that are commonly used in the NoSQL world.

In order to have some practical experience with NoSQL databases, in this chapter, we have gone through step-by-step tutorials of the popular frameworks Apache Cassandra, MongoDB, and Neo4j. We performed their step-by-step installation and basic configuration. We have also performed data manipulation, learning how to organize, store, and retrieve data from these database frameworks.

In the next chapter, we will explore the big data world over cloud. Along with that, we will also learn about one of the top commercially available cloud infrastructures, which is named Azure and managed by Microsoft.

5
Off-the-Shelf Commercial Tools

In Chapter 1, *Why Big Data*, we did various comparisons between on premises and cloud-based data centers. There are many companies in the race to provide better, secure, and low cost services for small to enterprise level companies, so that they can confidently start to utilize their services. Many offer pay as you go billing, helping avoid dealing with huge multi million startup costs.

In this chapter we will explore **Microsoft Azure,** formerly known as Windows Azure, which is provided by Microsoft, one of the market giants. It provides various services over cloud for all types of customers. It provides **Software as a Service** (**SaaS**), **Platform as a Service** (**PaaS**), and **Infrastructure as a Service** (**IaaS**). It supports many different programming languages, frameworks and tools, including Microsoft proprietary and third-party as well as open source software and systems. The primary comfort of using cloud infrastructure and services is that there is no startup cost and you can easily scale it up and down as per your requirements.

In this chapter, we will be covering the following topics with practical examples:

- Microsoft Azure and a list of the services it provides
- Build a streaming application with Azure that has the following components:
    - Setting up Microsoft Azure Account
    - Setting up Microsoft Azure Event Hub
    - Setting up Microsoft Azure Stream Analytics services
    - Create a Java application to stream data to Azure as a simulation of IoT device
    - Setting up Microsoft Power BI
    - Build a Dashboard with near to real-time statistics in Power BI

Let's get started with some more details about Microsoft Azure and what type of services it provides.

Microsoft Azure

As Windows Azure, the service was initially announced in October 2008 and was officially released on February 1, 2010. On March 25, 2014, it was renamed Microsoft Azure.

As of the time of writing, Azure offers 86 services. A few of these services are listed with a brief description of each:

- **Virtual Machines**: You can setup and launch a virtual machine of any operating system like Windows, Linux, and so on, in seconds.
- **App Services**: Publish your web and mobile apps over cloud with many predefined templates.
- **SQL Databases**: Use RDBMS database over cloud as a service.
- **Storage Services**: Provide scalable cloud storage as per your requirements.
- **Azure Cosmos DB**: Azure's own NoSQL database that supports multi data models and is scalable.
- **Azure Active Directory**: It can synchronize with on-premises directories and can work with single sign-on features.
- **Backup Services**: A **BaaS** solution that provides service to back up your on-premises data over cloud with geo-replication.
- **HDInsight**: A managed open source big data analytics service through which you can create clusters for Hadoop, Spark, Hive, HBase, and Kafka.
- **IoT Hub**: Can create, monitor, and manage billions of your IoT devices with bidirectional communication.
- **Content Delivery Network** (**CDN**): Microsoft have multiple datacenters all over the world that can back up CDN, for fast and secure delivery of contents and improved performance and availability.
- **Machine Learning Studio**: A browser based development and management tool for applying all the major machine learning algorithms, using a simple drag and drop interface.
- **Stream Analytics**: A service that can be use to analyze streams of data from multiple streaming sources, including IoT and non IoT devices.
- **Event Hubs**: A service that can be used to ingest streams of data from multiple sources with very low latency.

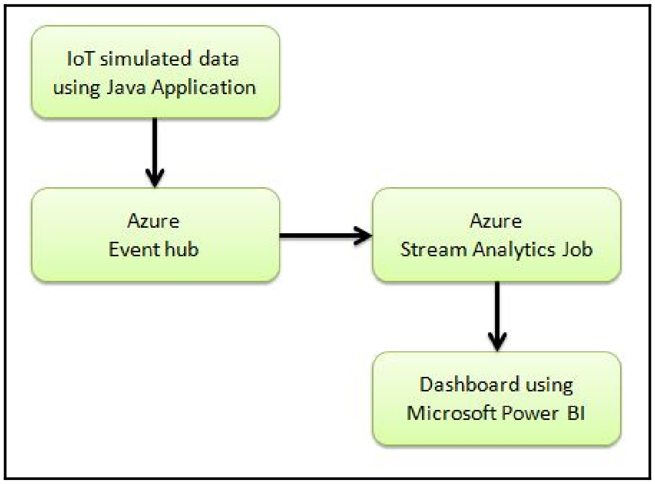
- **Data Lake Store**: It is built on the open HDFS standard to store unstructured, semi structured, and structured data, with no limits on the size. Data can be in terabytes or petabytes. It is highly scalable and can ingest trillions of objects.
- **Cognitive Services**: An AI service developed by Microsoft for developers to enhance their end user's experience by adding features such as image processing, speech recognition, searching capabilities, a recommendation system, and natural language processing.

These are but a few from the bunch of services that Microsoft Azure has to offer to its customers. All of these services can easily be scaled up and down as per user's requirements. Most of these services will charge on a pay as you go plan, which means that unless the user consumes processing and computation power through Azure, they will not be charged.

Now let's build an application utilizing the power of Microsoft Azure.

Building a practical application

The objective of this project is to send data from an IoT device over cloud and monitor its activities in a nice visual dashboard. For this purpose, we will be utilizing some of the Microsoft Azure services. The following figure illustrates the steps we will perform in this project:

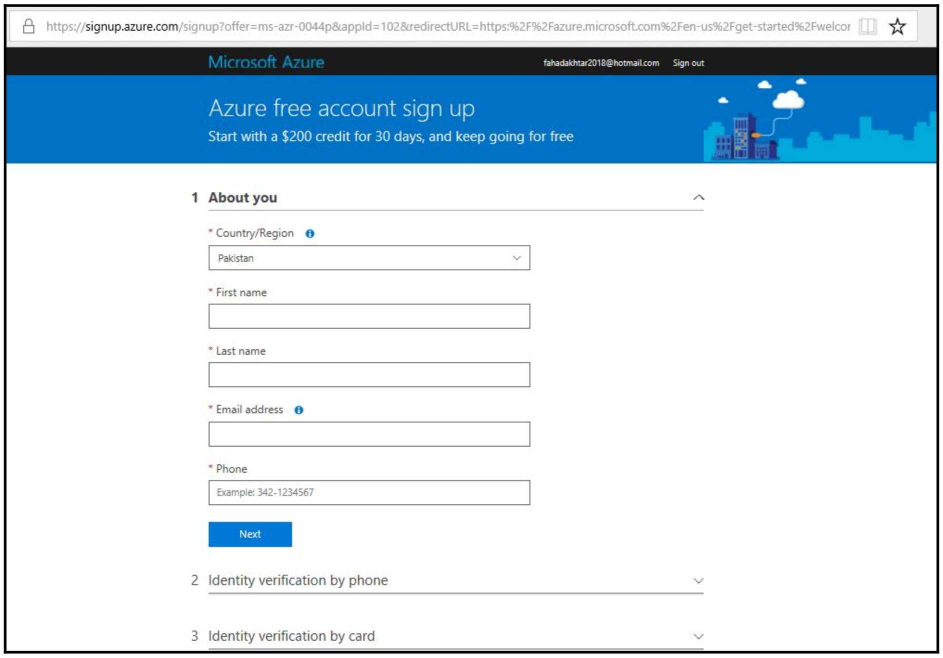

IoT simulated data using Java Application
Azure Event hub
Azure Stream Analytics Job
Dashboard using Microsoft Power BI

Let's get started by creating a Microsoft Azure account.

Microsoft Azure account

We will now create an account in Microsoft Azure. Microsoft are providing a free account for a year that allows use of their products with base line resources. They are also giving $200 credit with a free account for the first month, so that user can also test their services on an enterprise level without any limitations. Follow the following instructions to sign up for the service:

1. Point your browser to `https://azure.microsoft.com`.
2. Click on the free account signup link. It will redirect you to a login page. If you already have a Hotmail, Outlook, or MSN account, or any of your personal or enterprise domains configured in Office 365, you can directly log in. If not, click on the **Create One** link to take you to signup page. It will ask for some basic personal information and let you create your account.
3. Once you finish logging in or creating an account, you will arrive at the Azure sign up page, which asks for some basic information as shown in the following screenshot:



https://signup.azure.com/signup?offer=ms-azr-0044p&appId=102&redirectURL=https:%2F%2Fazure.microsoft.com%2Fen-us%2Fget-started%2Fwelcor

Microsoft Azure fahadakhtar2018@hotmail.com Sign out

Azure free account sign up

Start with a $200 credit for 30 days, and keep going for free

1 About you

* Country/Region: Pakistan

* First name

* Last name

* Email address

* Phone: Example: 342-1234567

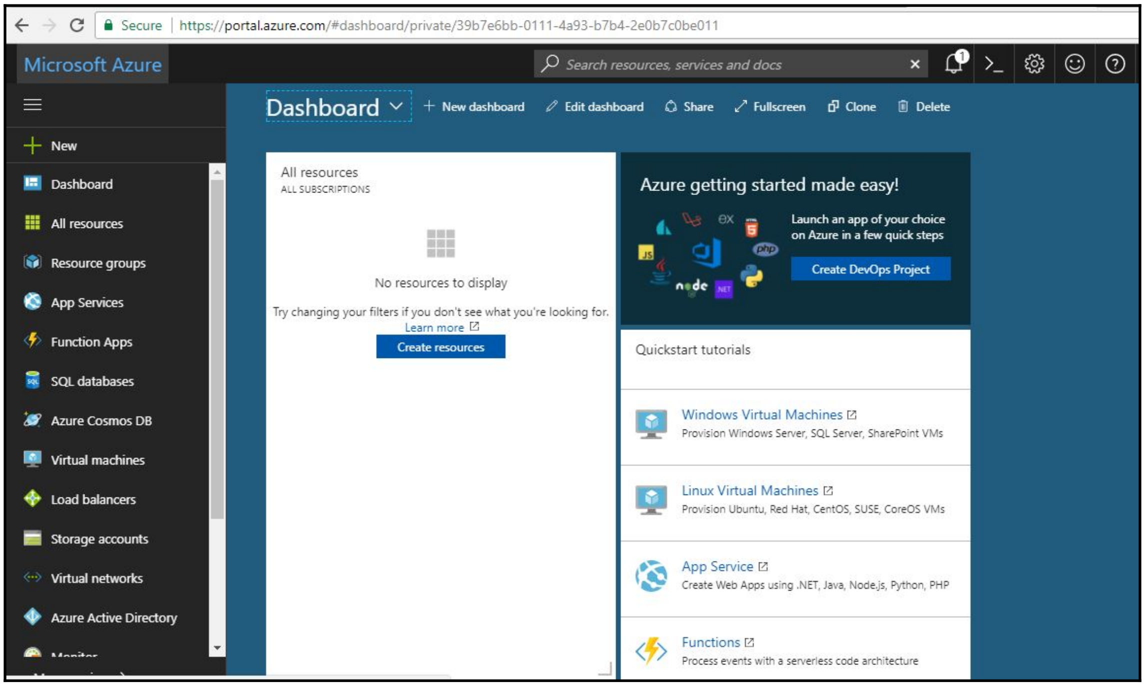
Next

2 Identity verification by phone

3 Identity verification by card

Sign up page for Azure

4. It will verify your identity by verifying your phone number, as well as requiring a valid credit card. Once you provide all the details, it will create your account and redirect you to the Azure dashboard, as shown in the following screenshot:


Secure | https://portal.azure.com/#dashboard/private/39b7e6bb-0111-4a93-b7b4-2e0b7c0be011
Microsoft Azure
Search resources, services and docs
New
Dashboard
All resources
Resource groups
App Services
Function Apps
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Dashboard
New dashboard
Edit dashboard
Share
Fullscreen
Clone
Delete
All resources
ALL SUBSCRIPTIONS
No resources to display
Try changing your filters if you don't see what you're looking for.
Learn more
Create resources
Azure getting started made easy!
Launch an app of your choice on Azure in a few quick steps
Create DevOps Project
Quickstart tutorials
Windows Virtual Machines
Provision Windows Server, SQL Server, SharePoint VMs
Linux Virtual Machines
Provision Ubuntu, Red Hat, CentOS, SUSE, CoreOS VMs
App Service
Create Web Apps using .NET, Java, Node.js, Python, PHP
Functions
Process events with a serverless code architecture

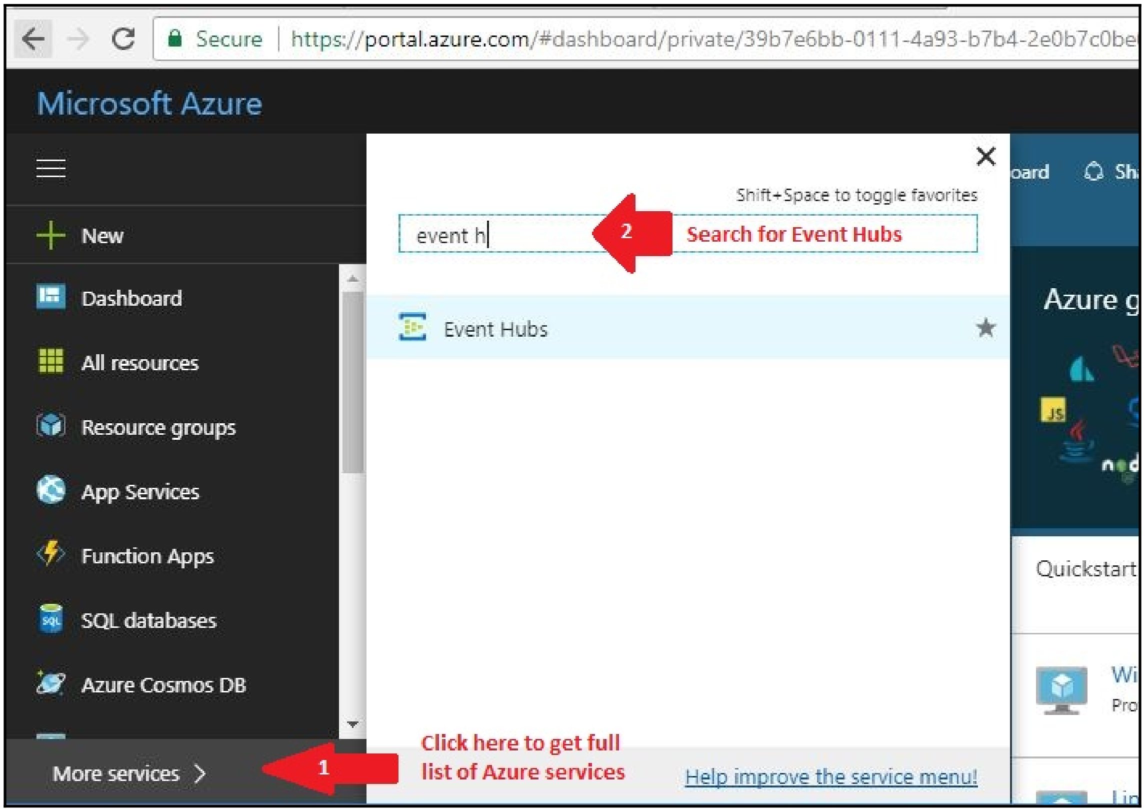
Dashboard for Azure

Now that we have setup our account in Microsoft Azure, in the next section, we will create an Event Hub.

The Azure Event Hub

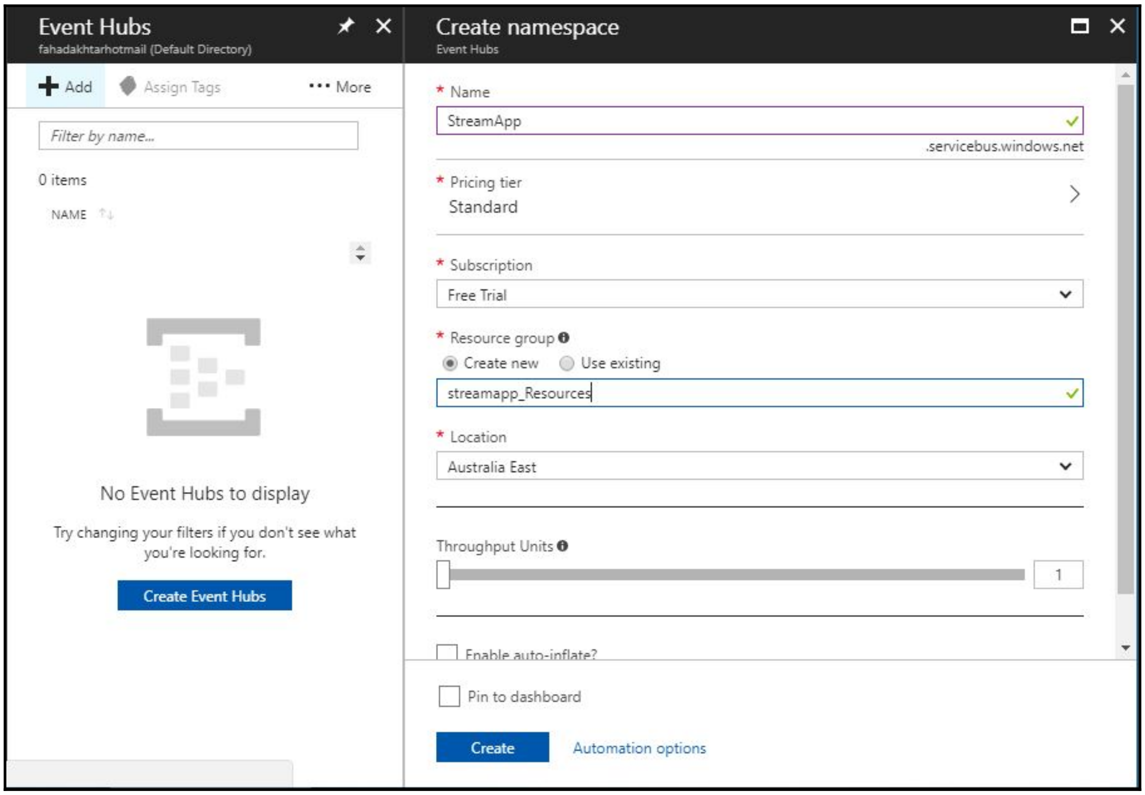
It's now time to create an **Event Hub**. One is required to receive data from IoT devices, which in our case will be a simulation application that will generate data similar to an IoT device. It will then be passed on to the Stream Data Analytics service:

1. Let's now open Azure Portal. On the bottom left side of the screen, click on the **More services** link and type **Event Hub** in the search box to look for the link, as shown in the following screenshot:


Secure
https://portal.azure.com/#dashboard/private/39b7e6bb-0111-4a93-b7b4-2e0b7c0be
Microsoft Azure
New
Dashboard
All resources
Resource groups
App Services
Function Apps
SQL databases
Azure Cosmos DB
More services
Shift+Space to toggle favorites
event h
2
Search for Event Hubs
Event Hubs
1
Click here to get full list of Azure services
Help improve the service menu!
Quickstart

Open an Event Hub service in Azure Portal

2. Once you open the **Event Hub** service, click on the **+Add** button from the top menu. It will display a form to fill in in order to create a namespace as shown in the following screenshot. The namespace is a container that will hold all the related elements in place:



* Name
StreamApp
.servicebus.windows.net

* Pricing tier
Standard

* Subscription
Free Trial

* Resource group
(● Create new) ○ Use existing
streamapp_Resources

* Location
Australia East

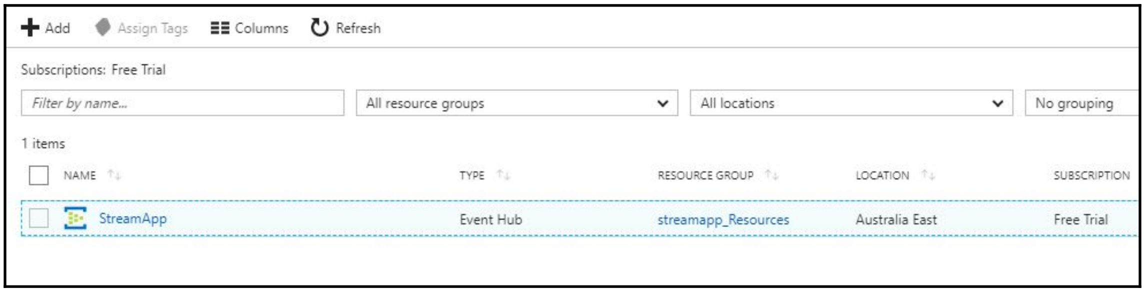
Throughput Units
1

☐ Enable auto-inflate?

☐ Pin to dashboard

Add Event Hub form

3. Once the namespace gets created, it will appear in the list, as shown in the following screenshot:



Add | Assign Tags | Columns | Refresh

Subscriptions: Free Trial

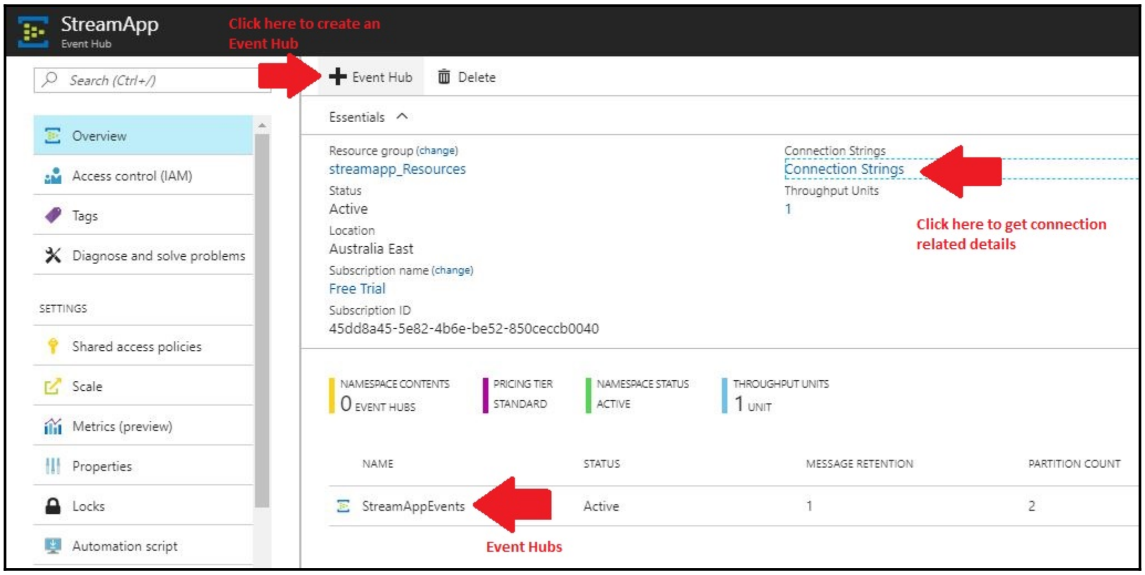
Filter by name... | All resource groups | All locations | No grouping

1 items

NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION
StreamApp	Event Hub	streamapp_Resources	Australia East	Free Trial

List of Event Hubs

4. Click on the newly created namespace **StreamApp** to open it. It will display a screen as shown in the following screenshot:



StreamApp
Event Hub

Click here to create an Event Hub

Search (Ctrl+/) | Event Hub | Delete

Essentials

Resource group (change)
streamapp_Resources
Status
Active
Location
Australia East
Subscription name (change)
Free Trial
Subscription ID
45dd8a45-5e82-4b6e-be52-850ceccb0040

Connection Strings
Connection Strings
Throughput Units
1

Click here to get connection related details

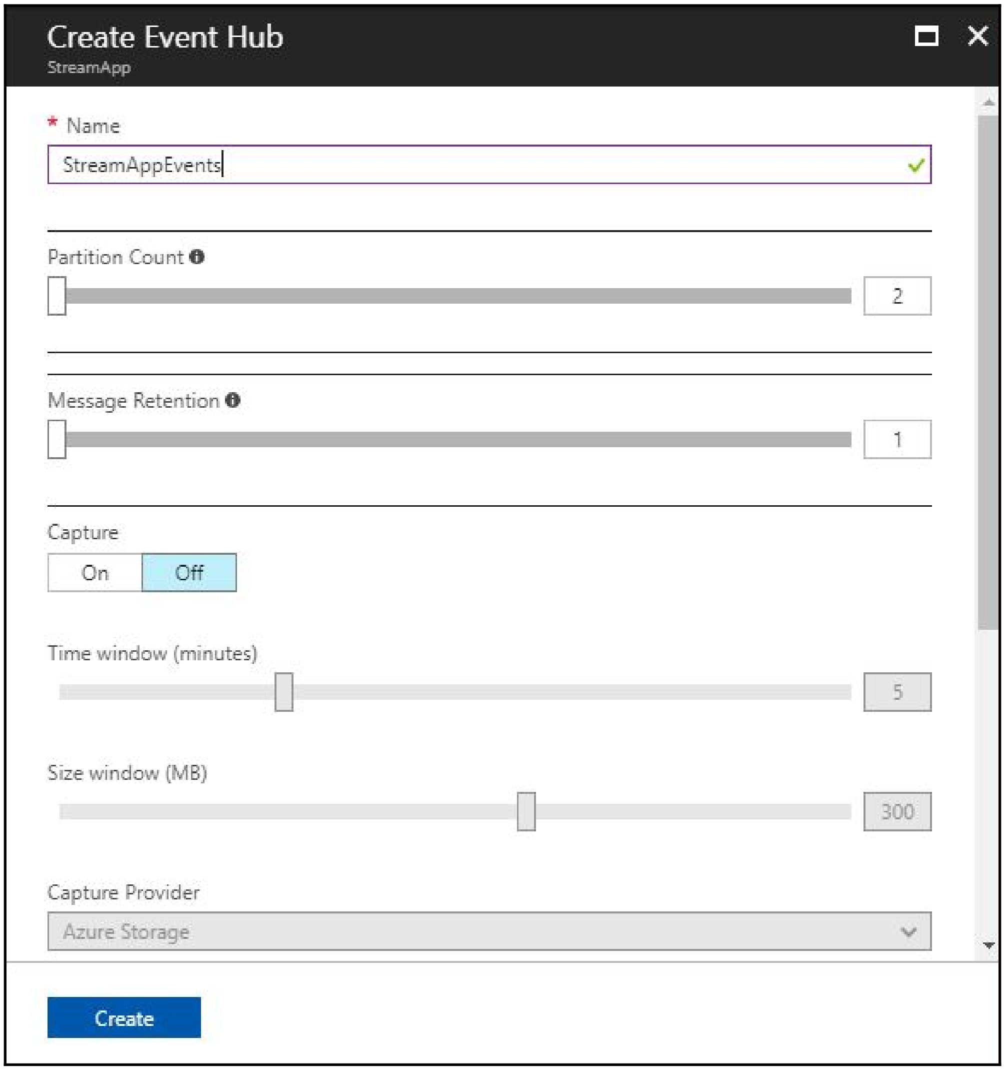
NAMESPACE CONTENTS 0 EVENT HUBS | PRICING TIER STANDARD | NAMESPACE STATUS ACTIVE | THROUGHPUT UNITS 1 UNIT

NAME	STATUS	MESSAGE RETENTION	PARTITION COUNT
StreamAppEvents	Active	1	2

Event Hubs

Event Hub Summary Page

5. Click on **+ Event Hub** to create an **Event Hub**, and fill in the required details as shown in the following screenshot:


Create Event Hub
StreamApp

* Name
StreamAppEvents

Partition Count: 2

Message Retention: 1

Capture: On [✓ Off]

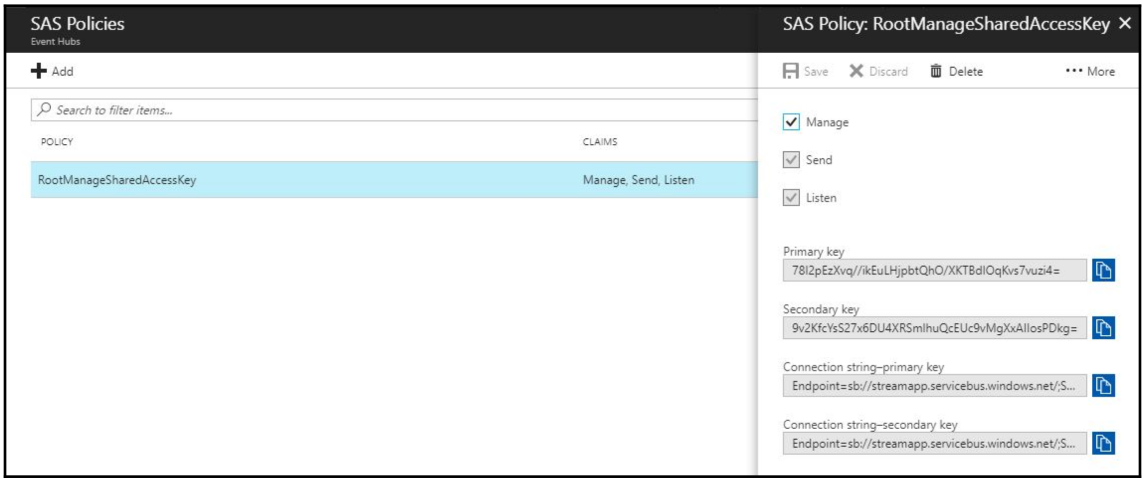
Time window (minutes): 5

Size window (MB): 300

Capture Provider: Azure Storage

Create Event Hub Form

6. Once the **Event Hub** has been created, it will appear in the list, as shown in the previous screenshot. Click on the **Connection Strings** on the right side of the page to open the details, as shown in the following screenshot:


SAS Policies
Event Hubs
Add
Search to filter items...
SAS Policy: RootManageSharedAccessKey
Save
Discard
Delete
More

POLICY	CLAIMS
RootManageSharedAccessKey	Manage, Send, Listen

✓ Manage
✓ Send
✓ Listen

Primary key
78I2pEzXvq//ikEuLHjpbtQhO/XKTBdlOqKvs7vuzi4=

Secondary key
9v2KfcYsS27x6DU4XRSmlhuQcEUc9vMgXxAllosPDkg=

Connection string–primary key
Endpoint=sb://streamapp.servicebus.windows.net/;S...

Connection string–secondary key
Endpoint=sb://streamapp.servicebus.windows.net/;S...

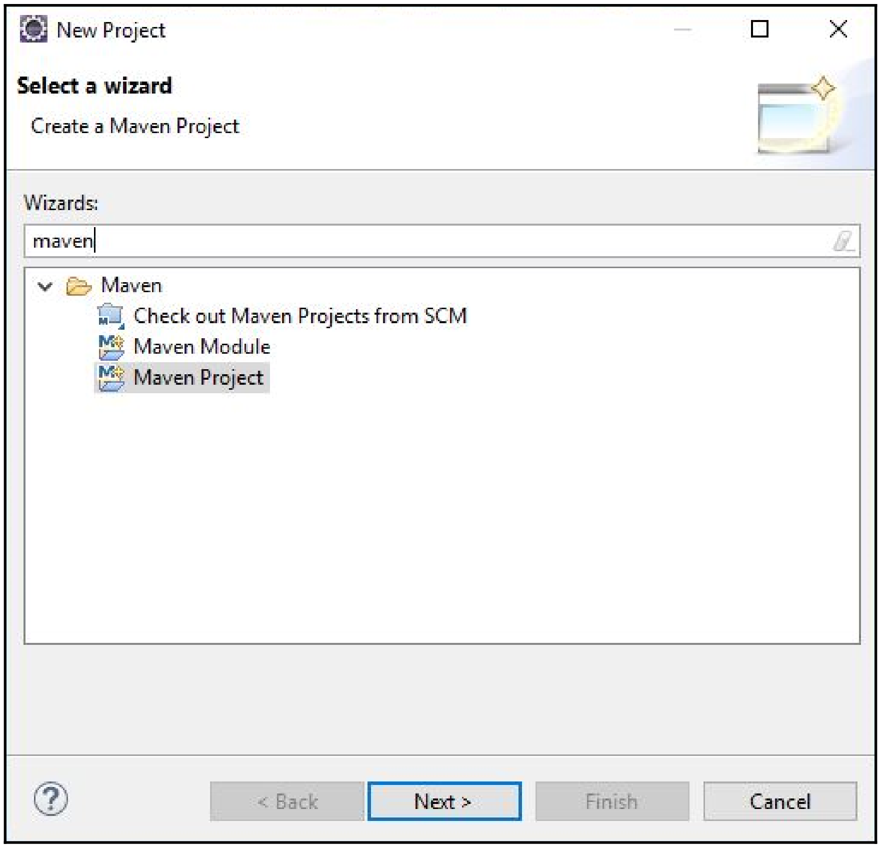
Connection String settings

The previous screen will display the list of access policies. By default, it will be created with the name `RootManageSharedAccessKey`. Once you click on it, it gives you the primary key and the secondary key along with their connection strings. We will need this information later to configure our simulation application to stream IoT data into the **Event Hub**.

IoT simulation application

Now that we have finished setting up Azure Event Hub to start receiving data from IoT devices, it's now time to build a Java application. It will simulate streaming data similar to an IoT device. We will be using Eclipse IDE for Java programming:

1. Open Eclipse. Select `New->Project` and select **Maven Project** as shown in the following screenshot:



New Project

Select a wizard

Create a Maven Project

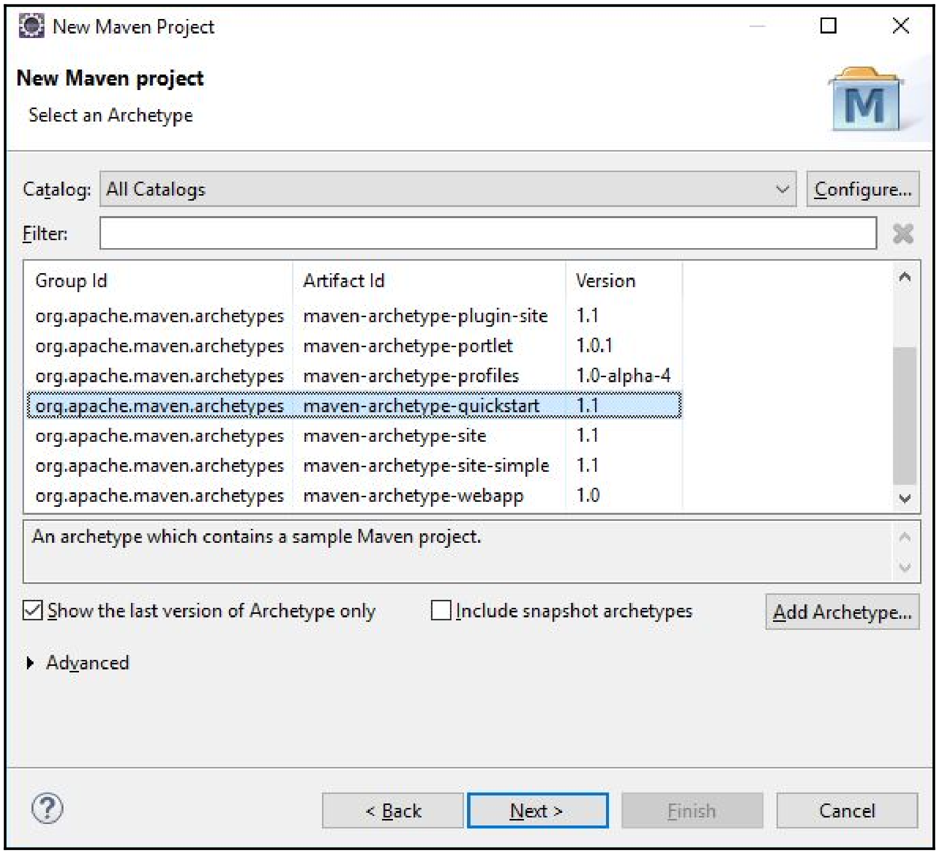
Wizards:

maven

Maven
- Check out Maven Projects from SCM
- Maven Module
- Maven Project

Create New Maven Project

2. Next step is to select the default workspace. If you want to keep the current workspace under which you started Eclipse, keep the default options selected and click **Next**. In the next step, it will ask you to select the **Group Id** and **Artifact Id** from the available list. Select the one with the **Artifact Id maven-archetype-quickstart**, as shown in the following screenshot:



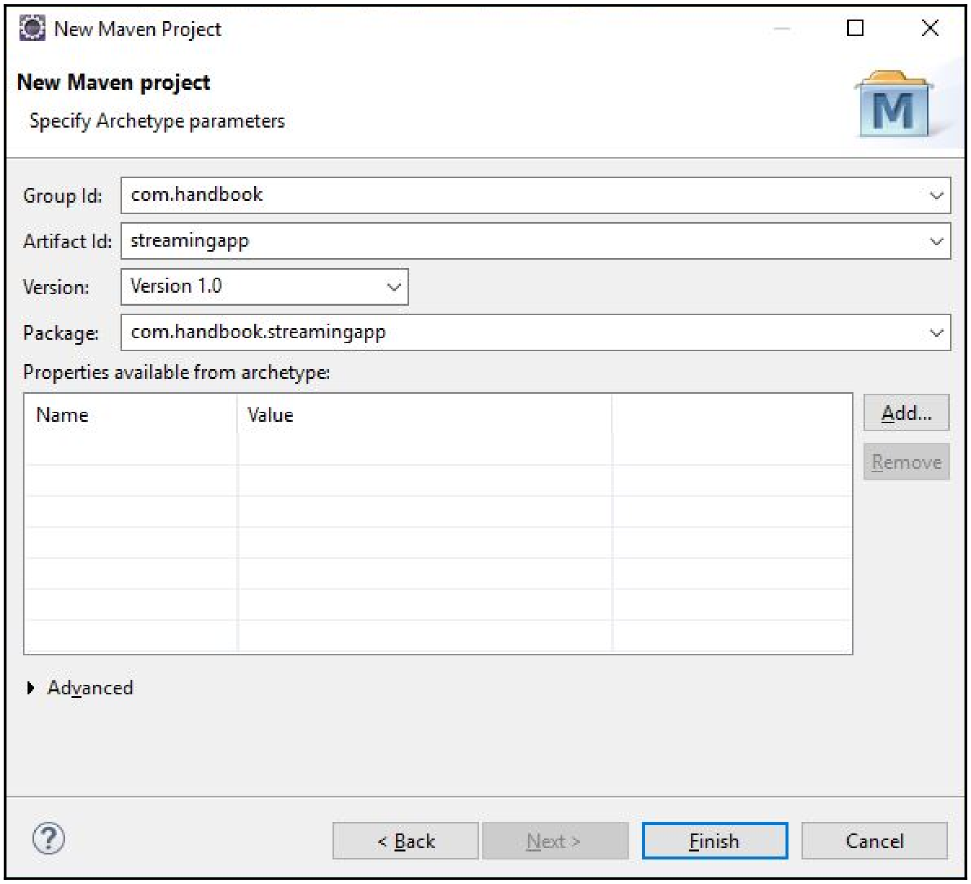
Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

An archetype which contains a sample Maven project.

[✓] Show the last version of Archetype only
[ ] Include snapshot archetypes

Maven Artifact selection

3. Now we have to define **Group Id** and **Artifact Id**. Here, you can name them as you like. In our case, they are defined as shown in the following screenshot:



Group Id: com.handbook
Artifact Id: streamingapp
Version: Version 1.0
Package: com.handbook.streamingapp
Properties available from archetype:

Name	Value

Specify Archetype Parameters

Now that the project has been created, it's time to define Maven Repository for Azure Eventhub. Copy the following lines of code in between the `<dependencies>` `</dependencies>` tag, in your `pom.xml` file, which is available at the root of your project, as shown in the following code:

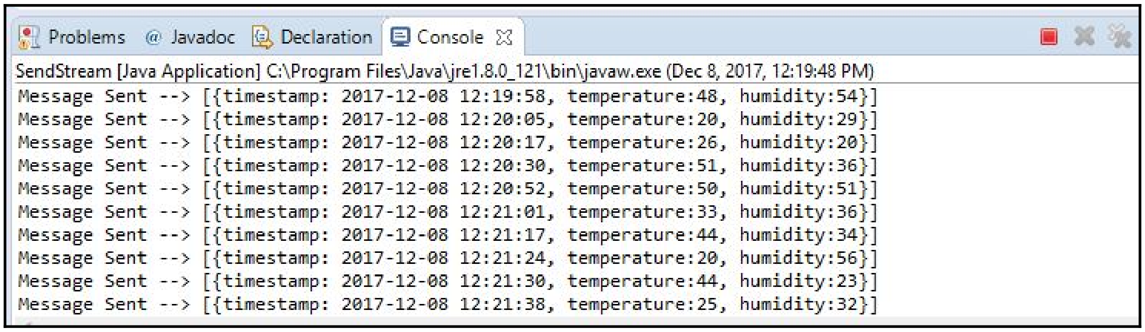
```
<dependency>
    <groupId>com.microsoft.azure</groupId>
    <artifactId>azure-eventhubs</artifactId>
    <version>0.15.1</version>
</dependency>
```

Click on **Save**, and give it a minute. Eclipse will automatically download the related `.jar` file for Azure Eventhub. Now create a new class called **SendStream** and write down the following code. It contains comments along each line to let you understand how it is working, as shown in the following code:

```
package com.handbook.streamingapp;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
import java.util.concurrent.ExecutionException;

import com.microsoft.azure.eventhubs.*;

public class SendStream {

  public static void main(String[] args)
      throws ExecutionException, InterruptedException, IOException,
EventHubException{
// fill in the required details below
// as per the variable name defined
      final String namespaceName = "StreamApp";
      final String eventHubName = "streamappevents";
      final String sasKeyName = "RootManageSharedAccessKey";
      final String sasKey = "----Put your shared key here---";

// define connection to connect to Azure Eventhib
      ConnectionStringBuilder connStr = new
ConnectionStringBuilder(namespaceName, eventHubName, sasKeyName, sasKey);
// timestamp format to pass as JSon
      SimpleDateFormat time = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

// Randomizer to generate random temperature and humidity values
// as simulation
      Random temp = new Random();
      Random humd = new Random();
// Infinite loop to continuously send data similar to an IoT device
      while(true) {
      int temperature = temp.nextInt(40)+20;
      int humidity = humd.nextInt(50)+20;
// Json string to pass to Azure eventhub
      String msg = "[{timestamp: "+ time.format(new Date()) +",
temperature:" + temperature + ", humidity:" + humidity + "}]";

// Convert the Json to byte array as EventData can only
// accept data in form of byte array
```

```
        byte[] payloadBytes = msg.getBytes("UTF-8");

    // EventData class initialize
        EventData sendEvent = new EventData(payloadBytes);

    // Eventhub Client class initialization based on the connection
    // string defined earlier
        EventHubClient ehClient =
    EventHubClient.createFromConnectionStringSync(connStr.toString());

    // Send data to Eventhub
        ehClient.sendSync(sendEvent);
    // Console output as to what was sent to the Azure
        System.out.println("Message Sent --> " + msg);

    // Wait for half second before sending the data again
        Thread.sleep(500);
        }
      }
    }
```

Right-click on the class and select **Run As | Java Application.** It will start sending the output to the **Console** window, similar to the one shown in the following screenshot:



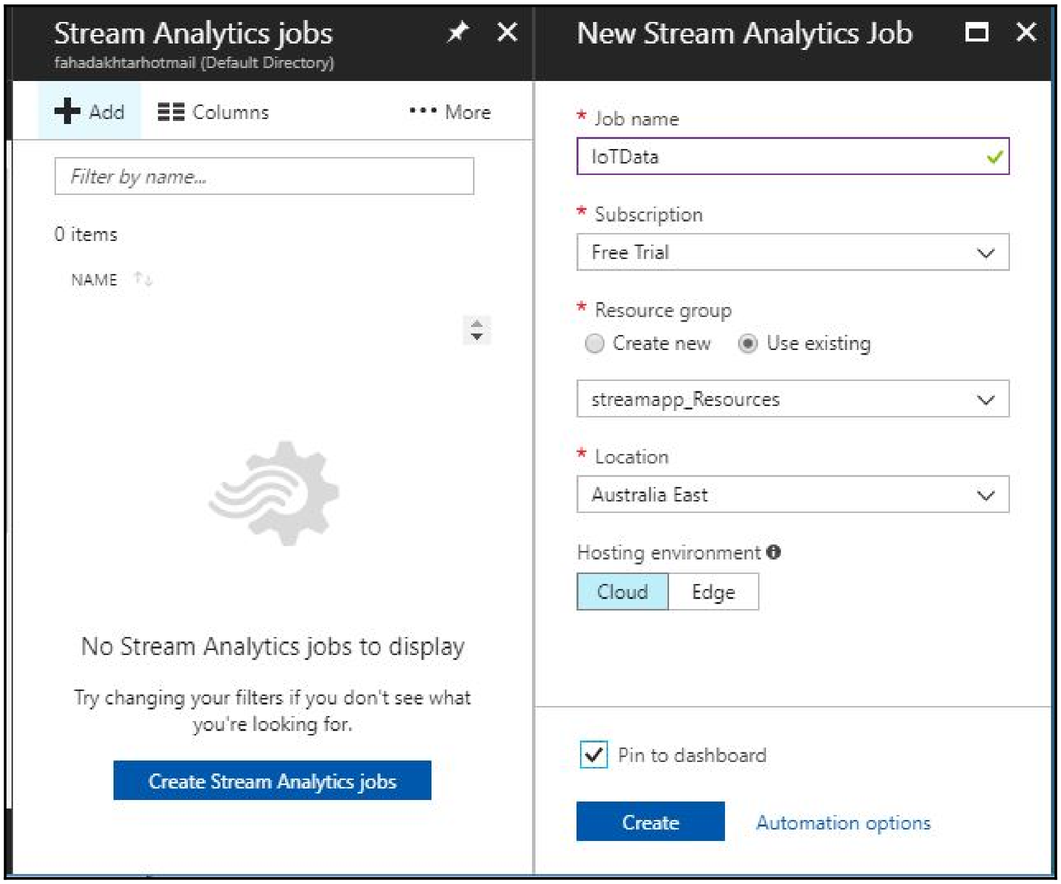
```
SendStream [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Dec 8, 2017, 12:19:48 PM)
Message Sent --> [{timestamp: 2017-12-08 12:19:58, temperature:48, humidity:54}]
Message Sent --> [{timestamp: 2017-12-08 12:20:05, temperature:20, humidity:29}]
Message Sent --> [{timestamp: 2017-12-08 12:20:17, temperature:26, humidity:20}]
Message Sent --> [{timestamp: 2017-12-08 12:20:30, temperature:51, humidity:36}]
Message Sent --> [{timestamp: 2017-12-08 12:20:52, temperature:50, humidity:51}]
Message Sent --> [{timestamp: 2017-12-08 12:21:01, temperature:33, humidity:36}]
Message Sent --> [{timestamp: 2017-12-08 12:21:17, temperature:44, humidity:34}]
Message Sent --> [{timestamp: 2017-12-08 12:21:24, temperature:20, humidity:56}]
Message Sent --> [{timestamp: 2017-12-08 12:21:30, temperature:44, humidity:23}]
Message Sent --> [{timestamp: 2017-12-08 12:21:38, temperature:25, humidity:32}]
```

Program Output Console

Here, we have finished developing a simulation application that will send data to Azure EventHub similar to an IoT device. It's now time to set up an Azure Stream Analytics job.

Setting up an Azure Stream Analytics job

Log into the Azure portal and click on **More Service** and search for **Stream Analytics Job**. Once you open it, click on **+Add**. It will display a form. Fill out the required fields as shown in the following screenshot:


Stream Analytics jobs
fahadakhtarhotmail (Default Directory)

0 items

No Stream Analytics jobs to display

Try changing your filters if you don't see what you're looking for.

New Stream Analytics Job

* Job name: IoTData

* Subscription: Free Trial

* Resource group: Create new [✓ Use existing]

streamapp_Resources

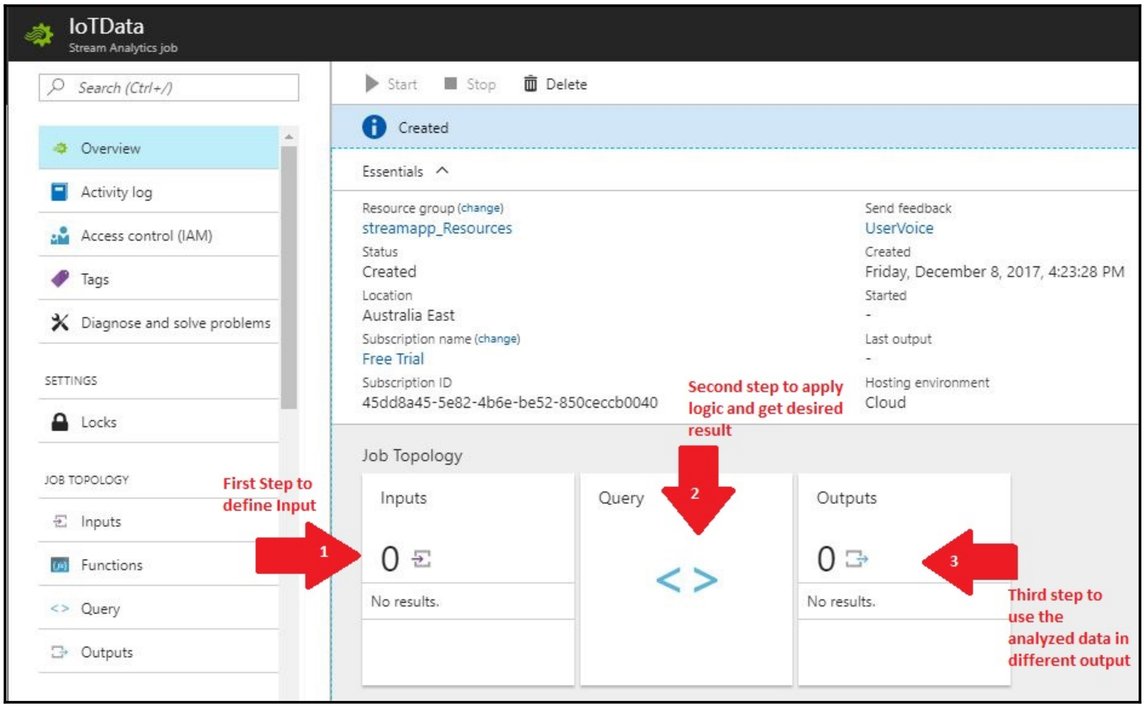
* Location: Australia East

Hosting environment: [✓ Cloud] Edge

[✓] Pin to dashboard

Create Stream Analytics Job

Once it gets created, it will be displayed in the list. Click on it to open the details and setup options as shown in the following screenshot:

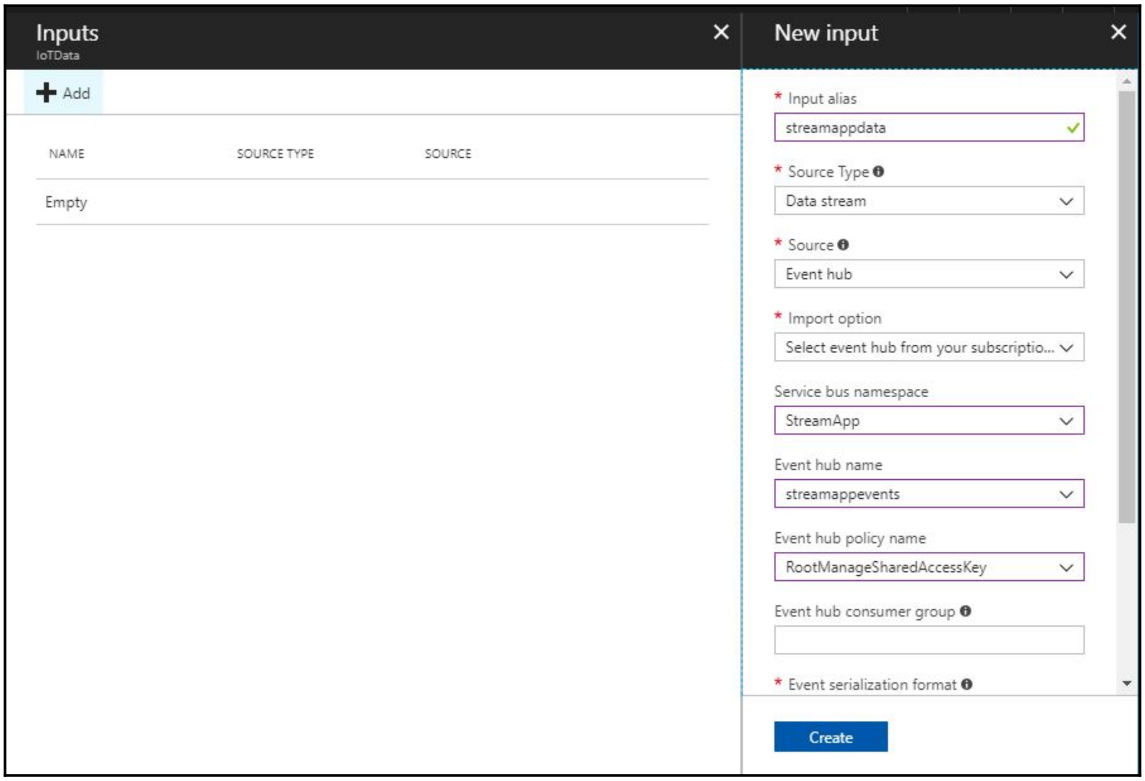


Resource group (change)
streamapp_Resources
Status
Created
Location
Australia East
Subscription name (change)
Free Trial
Subscription ID
45dd8a45-5e82-4b6e-be52-850ceccb0040
Send feedback
UserVoice
Created
Friday, December 8, 2017, 4:23:28 PM
Started
-
Last output
-
Hosting environment
Cloud

Stream Analytics Job Details

Input

In this screen, the first step is to set up the input to the Streaming Job. Inputs can be from many different types of sources, such as Event Hub, blob storage, and so on. To set up input, click on the **Inputs** box, marked as **1** in the previous screenshot. Once you click on it, it will display the list of configured inputs. Click on the **+Add** button to configure our first input. Once you click on the button, it will display a form to configure the input, as shown in the following screenshot:



* Input alias	streamappdata
* Source Type	Data stream
* Source	Event hub
* Import option	Select event hub from your subscriptio...
Service bus namespace	StreamApp
Event hub name	streamappevents
Event hub policy name	RootManageSharedAccessKey
Event hub consumer group	
* Event serialization format	

Configure Input for Stream Analytics Job

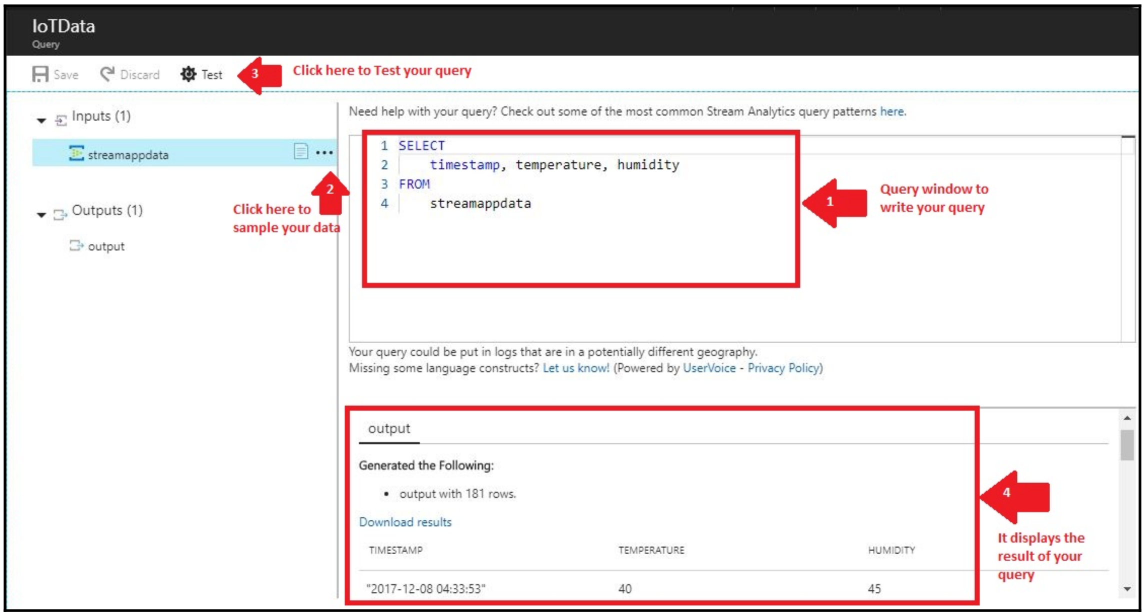
Give a name to your input; in our case it's **streamappdata**. Make sure the source type is selected as **Data stream** and the source is selected as **Event Hub**. In the Service bus namespace, select the one we created in the first step, **StreamApp**, the Event Hub name as **streamappevents**, and the Event Hub policy name as **RootManageSharedAccessKey**. If you have created and inputted all the options as mentioned here, your screen will look similar to the previous figure. Click on the **Create** button to create your input to the Stream Analytics Job.

Query

The second step is to execute a query to get the desired result from Inputs. In order to do so, click on the **Query** box. This will open up a query editor with which you can write your query and test for the desired result. For now, we will simply select the timestamp, temperature, and humidity from the data source, but if you want to write complex queries and apply formulas, you are welcome to do so. Our query is as follows:

```
SELECT
    timestamp, temperature, humidity
FROM
    streamappdata
```

Once you write the query, click on the **Test** button to run it against the sample data. To create a sample of data, you can either upload it or get it against your actual data. It can be done by clicking on the **...** button next to your data label on the left-hand-side side menu. The following screenshot illustrates these operations:


IoTData
Query
Save
Discard
Test
3
Click here to Test your query
Inputs (1)
streamappdata
2
Click here to sample your data
Outputs (1)
output
1
Query window to write your query
4
It displays the result of your query

Need help with your query? Check out some of the most common Stream Analytics query patterns here.

```
SELECT
    timestamp, temperature, humidity
FROM
    streamappdata
```

Your query could be put in logs that are in a potentially different geography.
Missing some language constructs? Let us know! (Powered by UserVoice - Privacy Policy)

output

Generated the Following:

- output with 181 rows.

Download results

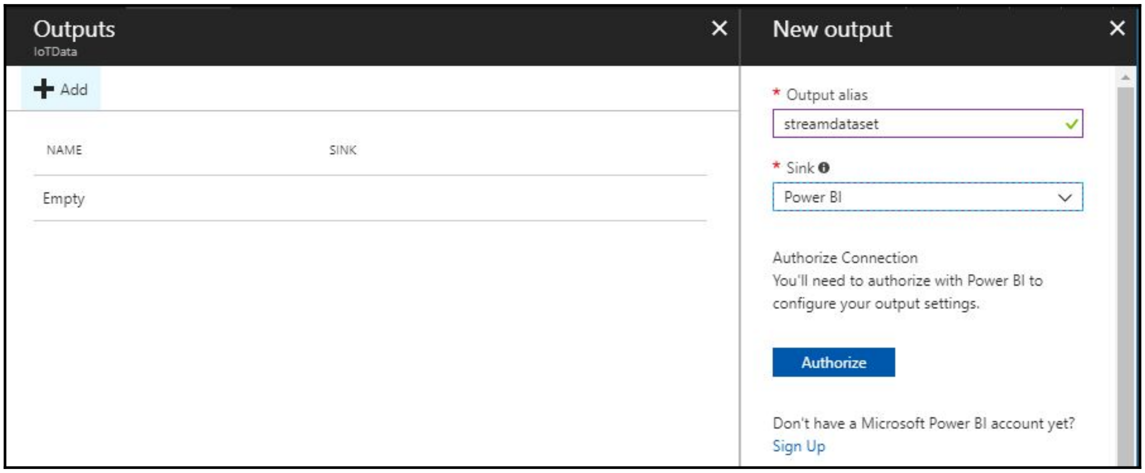
TIMESTAMP	TEMPERATURE	HUMIDITY
"2017-12-08 04:33:53"	40	45

Query Designer

Output

We have now tested our query and got the desired result. It's now time to configure output. Setting up an output means what we want to do with our result data; either we want to store it in a database, pass it on to some other Event Hub, or do some other visual or analytical job. Our objective is to create a visual representation as a dashboard to display near to real-time charts of temperature and humidity, as generated by an IoT. For this purpose, we will pass it on to **Microsoft Power Business Intelligence** (**PowerBI**).

Click on the **Output** box to set up what to do with the results. Once it opens, it will display a form. Give your output setting a name, like **streamdataset**. Select the sink type Power BI. Here, one thing to mention is that if you have never used PowerBI with Azure, it will ask you to authorize Azure to access the PowerBI workspace, as shown in the following screenshot. If you don't have PowerBI account, you can sign up for a 60-day free trial by clicking on the **Sign Up** link under the login form:



* Output alias: streamdataset

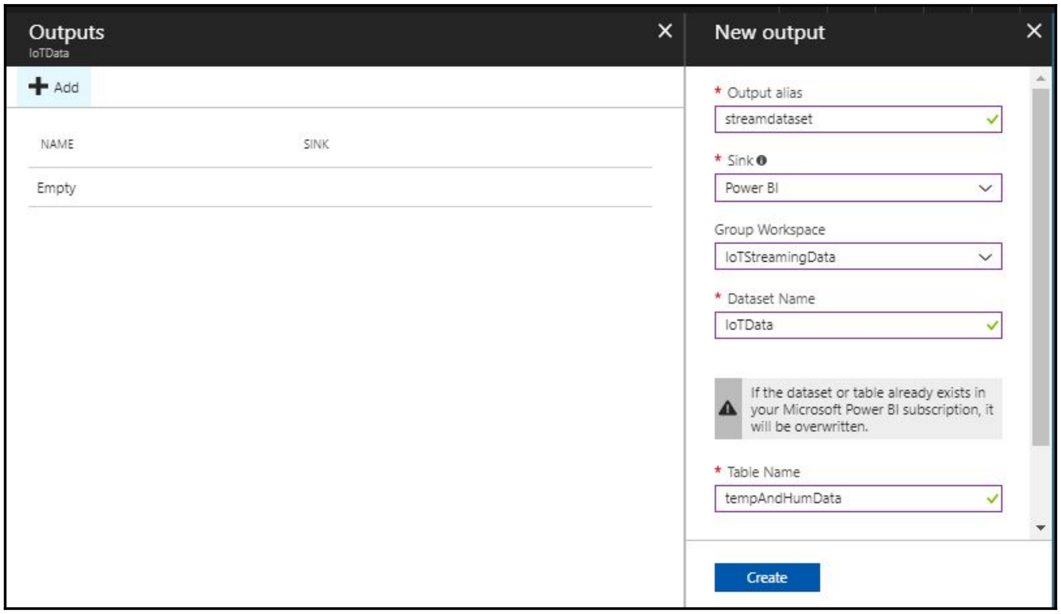
* Sink: Power BI

Authorize Connection
You'll need to authorize with Power BI to configure your output settings.

Don't have a Microsoft Power BI account yet?
Sign Up

Stream Analytic Job Output setup

Once you authorize it, you will see a list of all your workspaces available in PowerBI. I have already created a workspace named `IoTStreamingData`, which I will show the steps for in the next section. Give your **Dataset** and **Table** a name to use while creating a dashboard as shown in the following screenshot:



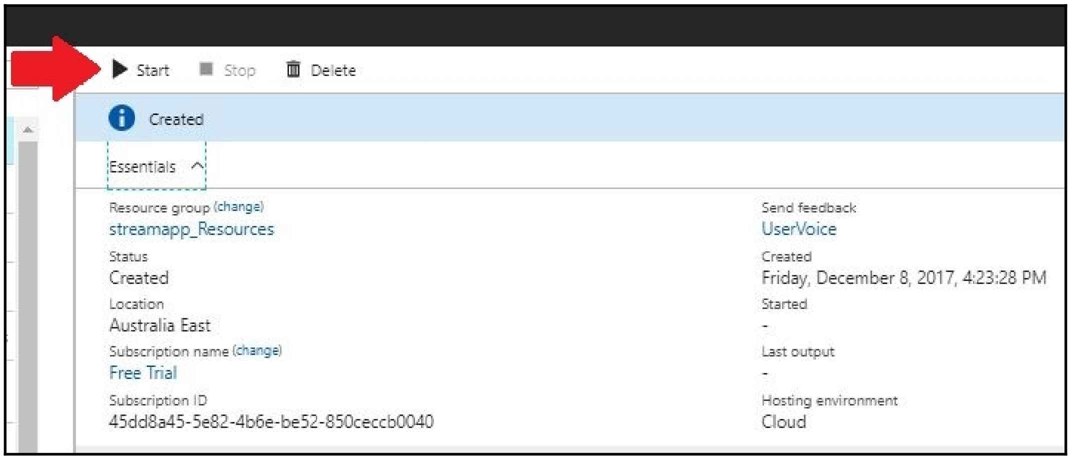
* Output alias: streamdataset
* Sink: Power BI
Group Workspace: IoTStreamingData
* Dataset Name: IoTData

If the dataset or table already exists in your Microsoft Power BI subscription, it will be overwritten.

* Table Name: tempAndHumData

Output setup

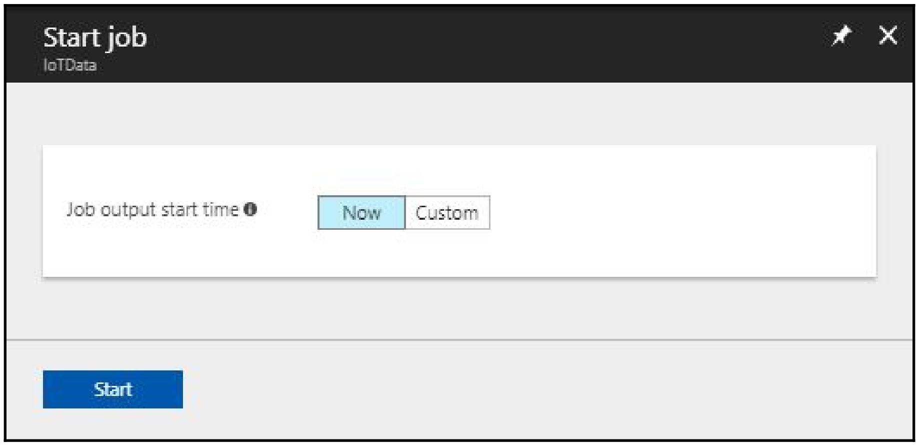
Click on the **Create** button to finish setting up the output. The last step is to start the newly created Stream Analytics Job. In order to do this, click on the **Start** button, as shown in the following screenshot:



Resource group (change): streamapp_Resources
Status: Created
Location: Australia East
Subscription name (change): Free Trial
Subscription ID: 45dd8a45-5e82-4b6e-be52-850ceccb0040
Send feedback: UserVoice
Created: Friday, December 8, 2017, 4:23:28 PM
Started: -
Last output: -
Hosting environment: Cloud

Start Stream Analytic Job

It will display a confirmation page that asks whether you want to start the service immediately or at some other custom time. In our case, we will start it immediately. Click on the **Now** button and then the **Start** button, as shown in the following screenshot:

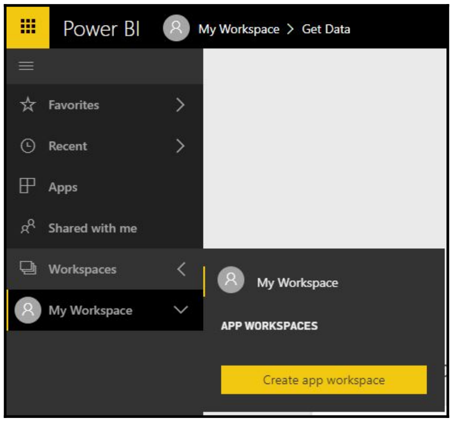


Confirmation to start service

It will take a minute or two to start the service. A notification will display on the top-right side once the service has started. Now, at this time, we have configured the input, query, and output relating to the Stream Analytics Job. It's now time create a dashboard in Power BI.

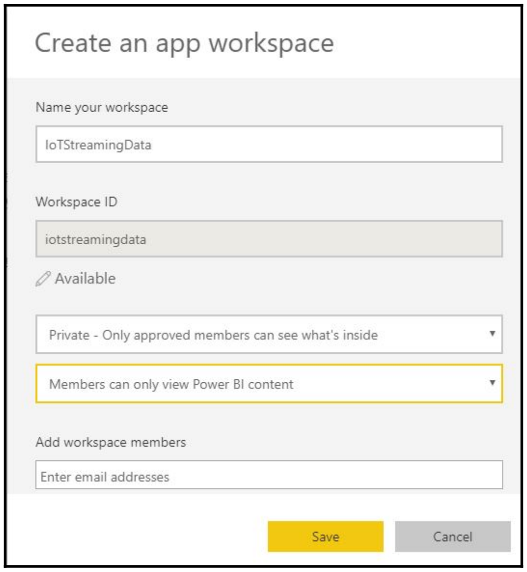
Dashboard in Power BI

We have come to a stage where we will now represent our data in a visual form; in the form of a dashboard. When you log in to Power BI for the first time, you have to create a Workspace. It is required for better organization, similar to what we do in application development. To do this, click on the **Workspace** menu in your left navigation and then click on **Create App Workspace** as shown in the following screenshot:



Creating Workspace in Power BI

Once you click on it, it will display a form for creating a workspace. Give it a name, select a privacy option as per your requirement, and click **Save**, as shown in the following screenshot. Once thing to notice here is that you have to create the workspace before you set up your Stream Analytic Job output, as it is one of its requirements for sending your data to Power BI. The Stream Analytic Job will then automatically create the dataset of the data it streams:


Create an app workspace

Name your workspace: IoTStreamingData

Workspace ID: iotstreamingdata

Available

Private - Only approved members can see what's inside

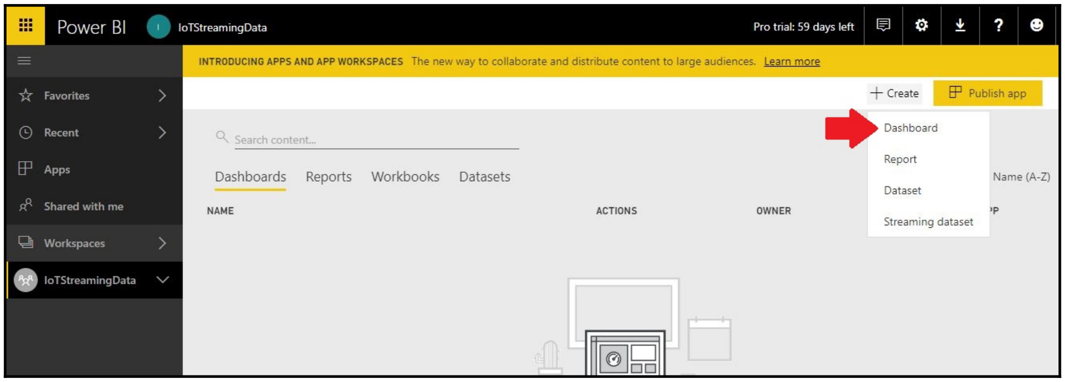
Members can only view Power BI content

Add workspace members: Enter email addresses

Save Cancel

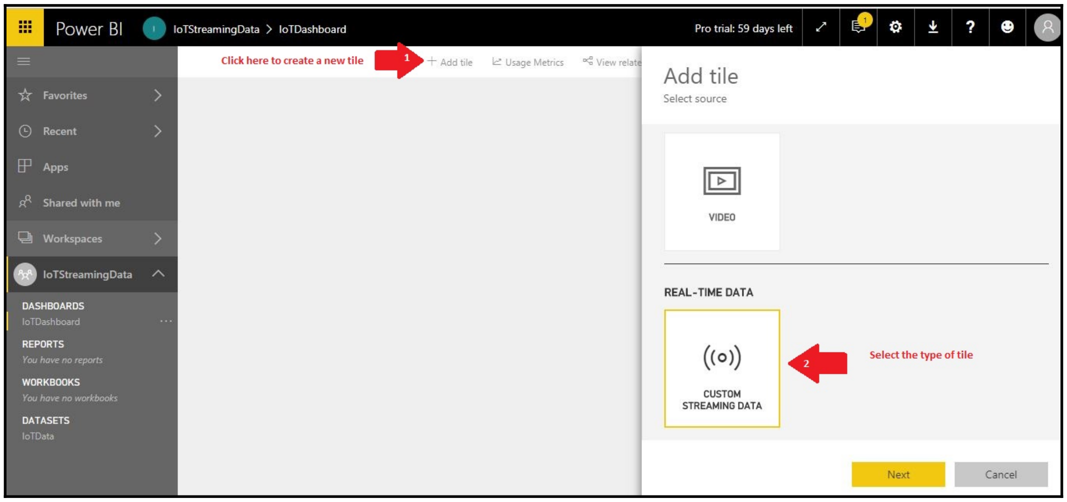
Power BI Workspace creation

Once your workspace gets created, it's now time to create a Dashboard. Click on the +**Create** button at the top right of the screen and select **Dashboard,** as shown in the following screenshot:



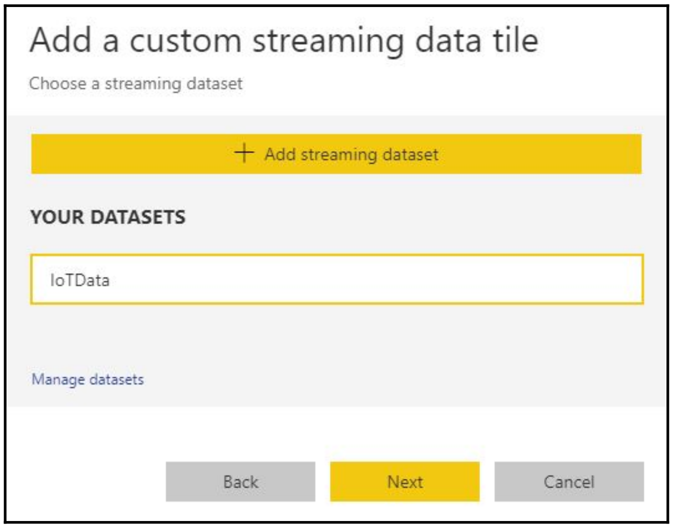
Create a Dashboard

It will pop-up a window for giving it a name. After inputting a name, click on the **Create** button to create your dashboard. Now that your dashboard has been created, it's time to add different tiles in it. Click on the **+Add tile** button in the top right corner. This will open a window; from it, select **Custom Streaming Data** from the **Real-Time** data category and click **Next**, as shown in the following screenshot:



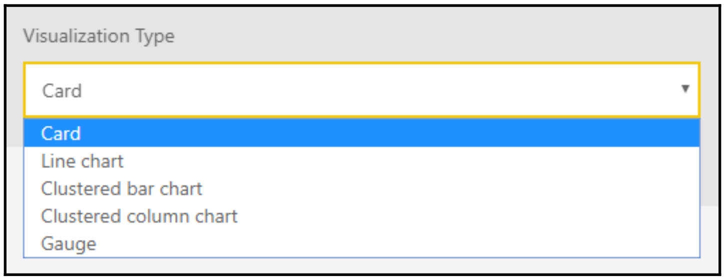
Creating a Tile in Dashboard

In the next window, you will be shown the list of datasets. Select the one we named while setting up the Stream Analytics Job, as shown in the following screenshot:



Selecting required dataset

On the next screen, you will be asked to select the type of tile. There are different type of tiles you can create for your dashboard, as shown in the following screenshot:



Type of tiles

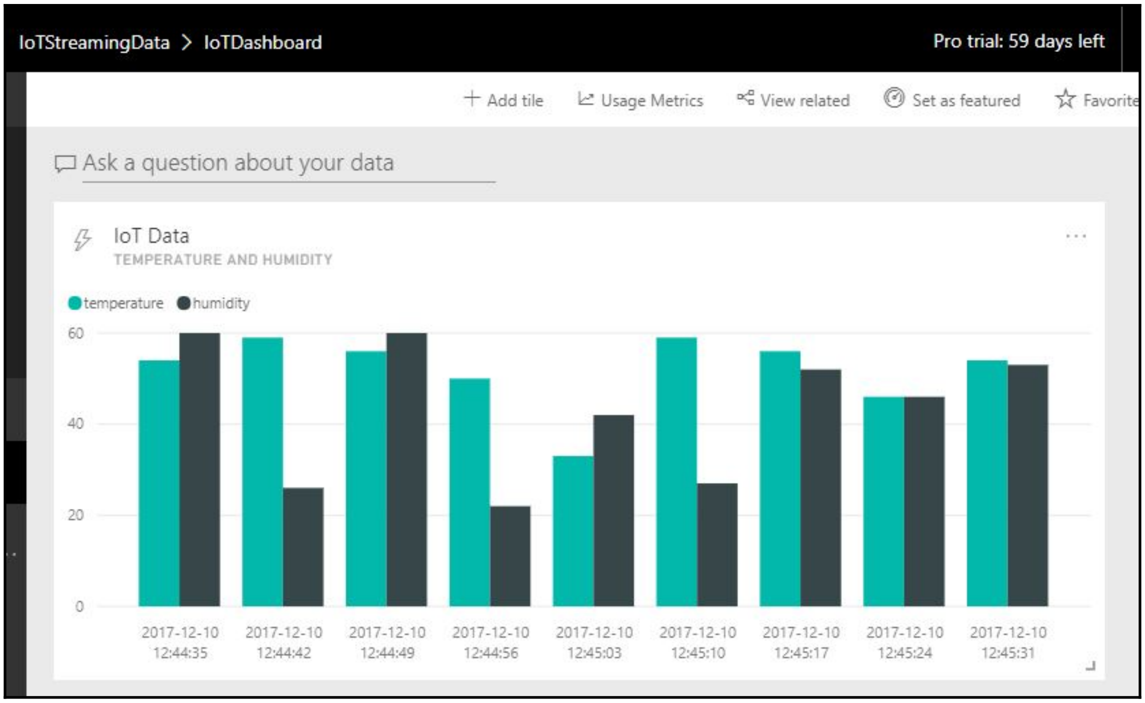
In our case, we have selected the following parameters:

- **Visualization Type** as **Clustered column chart**
- **Axis** will be **timestamp**
- **Value** will be **temperature** and **humidity**
- **Time window to display** will be **1 minute**

Having selected earlier, click the **Next** button and input the following:

- **Title** as **IoT Data**
- **Subtitle** as **Temperature and Humidity**

After inputting the previous details, click on the **Apply** button. The newly created tile will be shown in your dashboard, as shown the following screenshot:



IoT Dashboard with real-time data

We have successfully created our initial dashboard with simulated IoT data that produces **temperature** and **humidity** readings. You can add as many tiles with different types of representation as you like to make it more useful.

Summary

In this chapter, we have learned about one of the commercially available tools, Microsoft Azure, and the services it is offering for big data and cloud-related tools and technologies. Then, we have gone a bit further along the learning curve and performed a practical example of streaming data using Microsoft Azure services.

First of all, we created a free account in Microsoft Azure. Then we setup Azure Event Hub, which is responsible for handling the stream of data based on events. Then we created a Java application to simulated IoT data. We then set up a Stream Analytics Job where we configured inputs, a query to filter data as desired, and the output sink. We then created a Dashboard in Microsoft Power BI, which displays related information in a nice visual form.

In the next chapter, we will discuss and explore container-based virtualization. We will also explore related tools and technologies, such as Docker, Kubernetes, and so on.

6 Containerization

In a traditional system, an application usually deploys in a dedicated server. These servers are not cheap hardware and normally cost upwards of thousands of dollars. Deployment into a dedicated server is mainly done to provide full access to hardware resources and to avoid any configuration conflicts. However, this can cost companies a huge amount each time they're required to deploy a new application. From here, comes virtualization –although there are both advantages and disadvantages to its approach. To overcome the negatives of virtualization, containerization came into existence.

In this chapter, we will explore the following topics:

- Virtualization and different types of hypervisors
- Containerization and its benefits
- Docker
- Docker Swarm
- Kubernetes

We will start with the detailed theory related to these topics. We will then move our discussion onto something more practical, starting with installation. We will then perform some hands-on work using these frameworks. Let's get on and introduce virtualization.

Virtualization

In today's world, **virtual machines** (**VM**) are very common and are useful in many different scenarios. VMs help to create an isolated machine within your physical machine, with an operating system of your choice. There is no requirement of a specific OS from a given set. One of the advantages of a VM is that you can configure the underlying hardware as per your requirements; it can be of any architecture and you can add as many virtual devices as you want. You can create and run multiple virtual machines based on the available physical hardware. Although the physical hardware is shared among multiple VMs, in software and configuration terms, they are completely isolated from each other.

Hypervisors

In the virtualization world, there is an important component called a **hypervisor**. You can think of it as a hardware traffic monitor and resource allocator. It shares the host hardware's resources among different guest VMs, such as its CPU, memory, and network. Hypervisors help to ensure a better utilization of hardware resources by allowing the hosting of multiple instances on the guest OS.

A hypervisor isolates all guest VMs from each other, so they don't have any knowledge of each other. If any of the guest VMs are affected by malware or fall victim to an outside attack, the other VMs will not be affected. This makes this kind of technology extremely secure.

One of the key benefits of a hypervisor is that it makes underlying hardware details irrelevant to the guest VM, which makes the migration of VMs very easy. Whether you require your VM to move locally or to a remote location, there will be no hiccups as long as the destination machine has the required hardware resources. Here, the migration will be smooth and the migrated VMs can get to work immediately.

There are two types of hypervisors available: hardware-based and software-based. We will discuss these types in the following section.

Hardware-based hypervisors

This is also referred to as a **Type 1 hypervisor**. This runs directly on the host hardware, and is also known as a native, embedded, or bare metal hypervisor. It is a part of the hardware firmware and works closely with it in allocating hardware resources to different OSs. Examples of these types of hypervisors include Hyper-V, Oracle OVM, VMWare, and ESXi.

Software-based hypervisors

This is also referred to as a **Type 2 hypervisor**. This runs on top of the host operating system as a software package. It works as a **Virtual Machine Manager** and allocates different hardware resources using calls made to the host operating system. Examples of these types of hypervisors are Oracle VirtualBox and VMware Fusion.

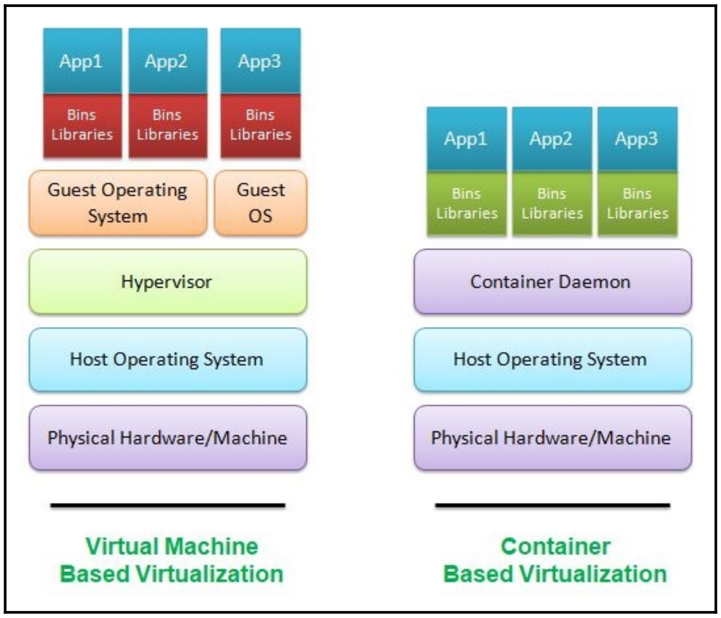
Now that we understand how virtualization works, let's proceed and look at containerization.

What is containerization?

Virtual machines are of great importance, and although we cannot work without them, there are some areas in which virtual machines are lagging behind. For example, if you want to run a single application in an isolated environment, you have to set up and run a full operating system before you're able to do so. This will have its own boot up time and will consume a lot of resources when loading its binaries and libraries. In turn, this will limit the number of VMs you can run on a single physical machine.

So, what is containerization? Containerization is similar to a virtual machine which runs in an isolated environment without consuming a huge amount of system resources. Instead of loading a full operating system, containerization uses the base kernel of a required operation system and loads only the libraries required by our desired application. This will take seconds to load, whereas VMs can sometimes take minutes to start. Due to their lightweight nature, containers let you start more isolated applications in a single physical machine.

The following diagram illustrates the architectural comparison of applications running in a virtual machine and in a container:



Virtual machine and container-based architecture

Now we will discuss the benefits of using containers over virtual machines.

Benefits of containers

Let's explore a few of the key benefits of using containers. They are as follows:

- Containers use fewer resources, such as CPU and memory, compared to VMs, as they do not require a full operating system to load
- You can start, stop, and destroy containers in seconds
- Containers are very portable; once you set up a container, it can be run on any environment without additional configuration or setup
- You can easily perform the horizontal scaling of containers in a clustered environment
- Containers consume less storage space compared to VMs, as they do not require the binaries and libraries of a full operating system

There are different container daemons and applications available on the market for you to adopt, such as Docker, Kubernetes, CoreOS rkt, Apache Mesos, and Canonical LXD. We will be discussing Docker and Kubernetes in the following sections.

Docker

Docker is an open source software technology managed by Docker Inc. It was initially released in March 2013. Docker makes it easy for you to create, run, and deploy containers. It is designed to be part of an application's development and deployment lifecycle, leaving developers and system administrators reassured that it will run smoothly on any machine without omitting any dependency or configuration. Now, let's explore what happens when you try to run a Docker image.

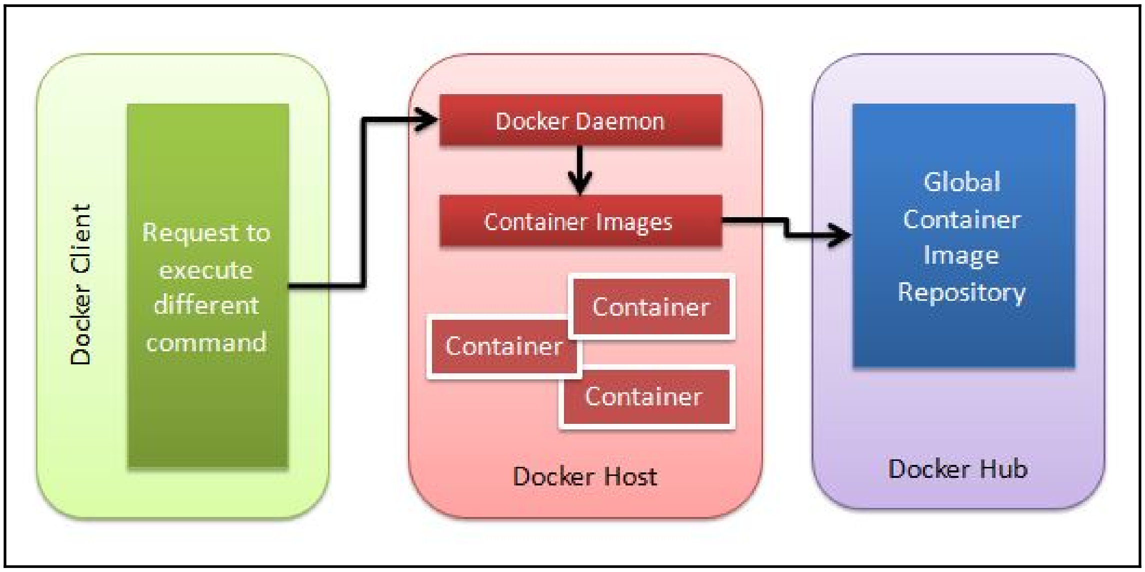
Docker workflow

There are three components in a Docker engine; they are as follows:

- **Docker Client**: As the name states, this is a client on the front end that is used to give commands to the Docker host; these commands can be of any type.
- **Docker Host**: This is the main engine that stores container images and runs them. It executes different commands as requested by the Docker client.

- **Docker Hub**: This is a repository of container images. Once a client requests to run an image, Docker Host checks internally whether the image exists locally or not. If the image doesn't exist, it pulls the image from the Docker Hub.

The following diagram illustrates the workflow of Docker:



Docker workflow

There are two editions available for Docker, and they are as follows:

- **Docker Enterprise Edition** (**EE**): This is based on a subscription-based mechanism and includes support with SLAs. It has different subscription tiers, including an integrated Docker Datacenter, security, and clusters.
- **Docker Community Edition** (**CE**): This is freely available for anyone to use. Docker releases a stable release of this edition on a quarterly basis and an edge release on a monthly basis.

Installation

Now that we have developed an understanding of containers and Docker, let's install and experience them first-hand. Follow the given steps to install Docker Community Edition in Ubuntu Linux.

Execute the following command to enable the capability of adding a repository over the `http` command:

```
$ sudo apt-get install apt-transport-https ca-certificates curl software-
properties-common
```

Execute the following command to add the Docker official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
```

Execute the following command to add the Docker repository:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Now, update your local repository database by executing the following command:

```
$ sudo apt-get update
```

It's now time to install Docker Community Edition into your system; do this by executing the following command:

```
$ sudo apt-get install docker-ce
```

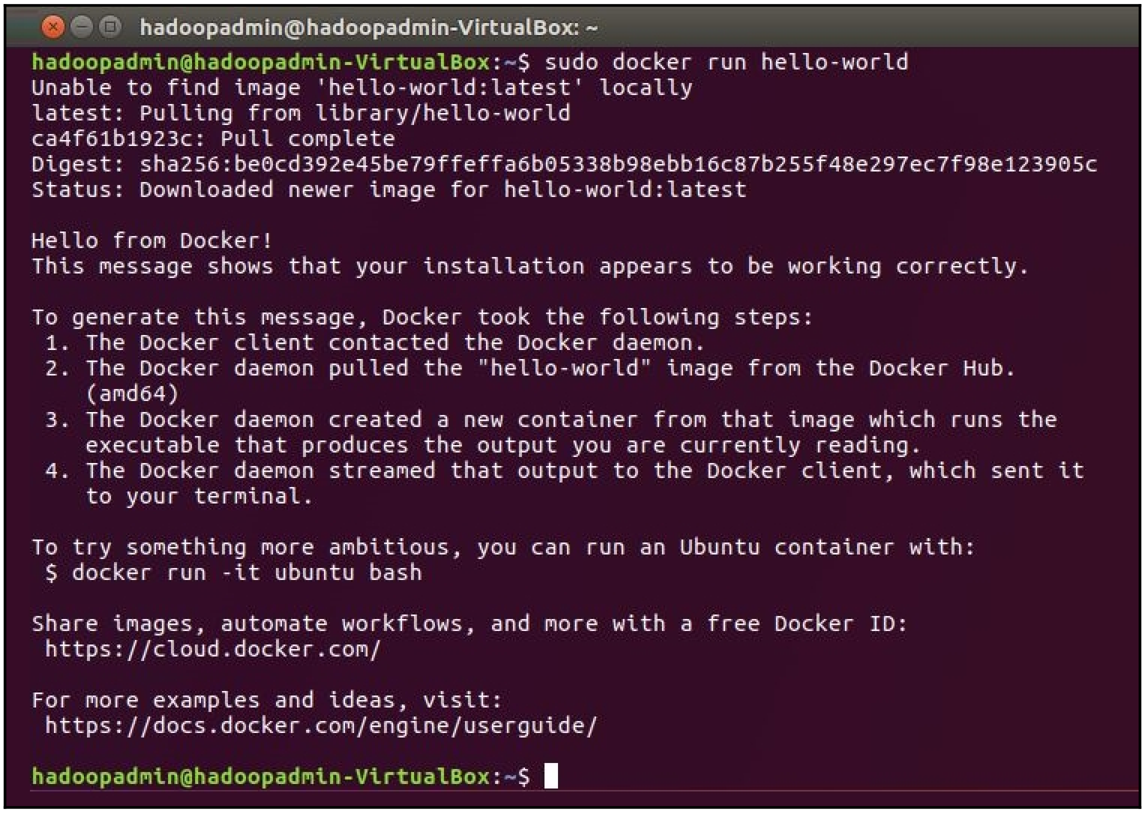
This command will install the highest available stable version of Docker on your system. If you want to install a specific version, use the following command. Make sure to replace `<VERSION>` with the required version:

```
$ sudo apt-get install docker-ce=<VERSION>
```

The download is around 129 MB and takes a few minutes to install. Once it's installed, execute the following command to verify the installation:

```
$ sudo docker run hello-world
```

Docker has provided a `hello-world` image to test that everything is installed correctly in your system and that you have the capability to execute containers. This command will download a `hello-world` image from the Docker server and will run in a containerized environment with a welcome message from Docker, as shown in the following screenshot:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:be0cd392e45be79ffeffa6b05338b98ebb16c87b255f48e297ec7f98e123905c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

hadoopadmin@hadoopadmin-VirtualBox:~$ 
```

Docker 'hello-world' image

Basic commands

Let's explore some of the basic commands in Docker and execute them in the Terminal. This will help us get an overview of the current Docker installation and the containers available in the server. First, let's start with the `help` command. Similar to other applications, you can use the `help` postfix to get different arguments that the `docker` command supports. Execute the following command to list the `help` menu:

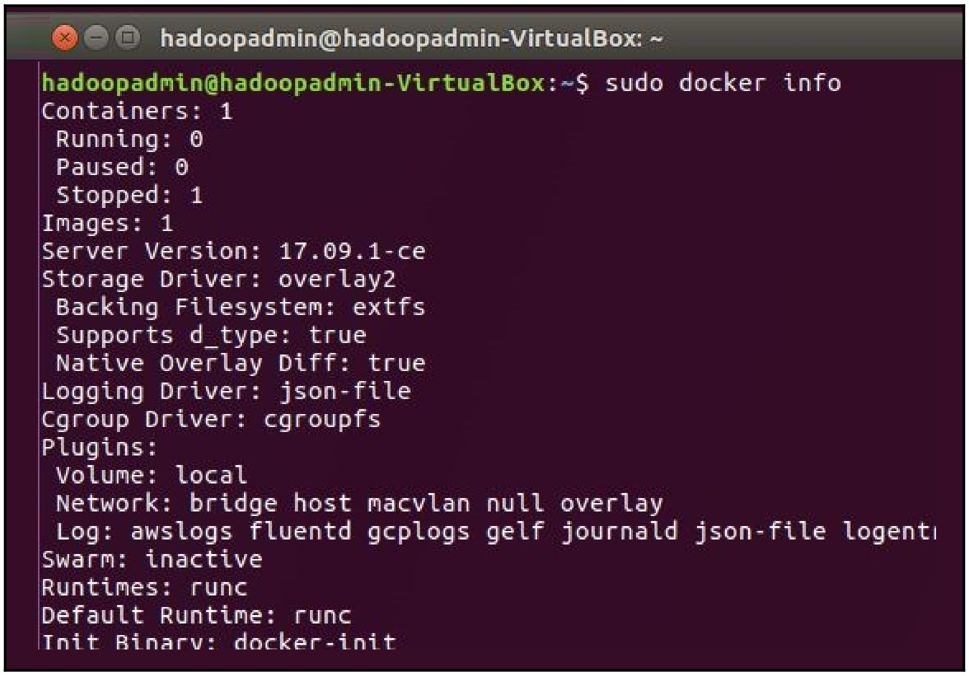
```
$ docker --help
```

In order to identify the currently installed `version` of Docker, execute the following command:

```
$ docker --version
```

If you want to find out detailed information about Docker, including how many container images are currently installed, how many of them are in different running states, and so on, execute the following command. The output of this command is shown in the following screenshot:

```
$ sudo docker info
```



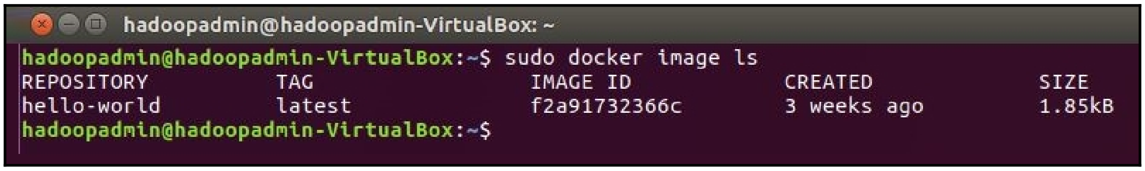
```
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker info
Containers: 1
 Running: 0
 Paused: 0
 Stopped: 1
Images: 1
Server Version: 17.09.1-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logent
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
```

Docker detailed information

You can see in the previous screenshot that only one image and container is currently available, and it is in the `stop` state. It is the `hello-world` image that we downloaded from the Docker server to test our installation. Once you start creating or downloading different images and begin to run them, this counter will increase.

If you want to find out what images or containers are available in your local machine, execute the following command to list all of them. The command's output is illustrated in the following screenshot:

```
$ sudo docker image ls
```

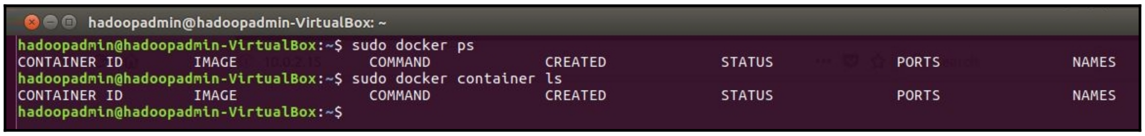


```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              f2a91732366c        3 weeks ago         1.85kB
hadoopadmin@hadoopadmin-VirtualBox:~$
```

List of locally available images and containers

To list the containers that are currently running, execute the following command. The screenshot following the command illustrates its output:

```
$ sudo docker ps

// Alternate command to list all running containers
$ sudo docker container ls
```



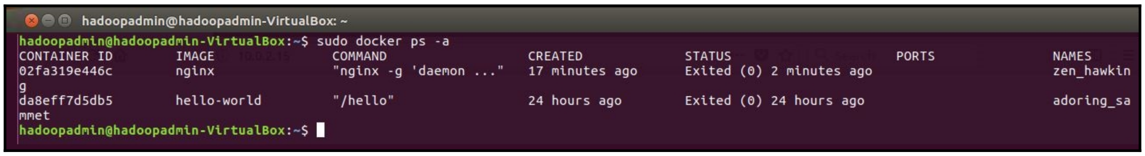
```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
hadoopadmin@hadoopadmin-VirtualBox:~$
```

List of currently running containers

To avoid using the `sudo` prefix in each `docker` command, add your local user account to the Docker group.

To list every container, whether it's running or in a stop state, execute the following command. The screenshot following the command displays its output:

```
$ sudo docker ps -a
```



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                     PORTS               NAMES
02fa319e446c        nginx               "nginx -g 'daemon ..."   17 minutes ago      Exited (0) 2 minutes ago                       zen_hawkin
g
da8eff7d5db5        hello-world         "/hello"                 24 hours ago        Exited (0) 24 hours ago                        adoring_sa
mmet
hadoopadmin@hadoopadmin-VirtualBox:~$
```

List of all containers

To stop a running container, execute the following command:

```
// Replace adoring-sammet with the container name you want to stop
$ sudo docker stop adoring-sammet
```

To remove or delete a container image, execute the following command. Remember that you must first make sure that the image is not running and is in the stop state:

```
// Replace adoring-sammet with the container name you want to delete
$ sudo docker rm adoring-sammet
```

In order to see the different resource utilization of containers, such as CPU and memory, execute the following command:

```
$ sudo docker container stats
```

Now that we have gained some hands-on experience of basic `docker` commands, let's now explore how we can make our own Docker images and then run those images as containers.

Docker images

At this stage, we should have successfully installed Docker and tested some basic commands over the CLI to manage Docker. Now, let's move on to the next stage and create our own Docker image and a container from it. The purpose of this Docker build is to host a static HTML page on the NGINX web server, which runs the Alpine Linux distribution under the hood.

Building a Docker image

Let's get started. The first step is to create a folder named `dockerImage` by executing the following command. This will help to give you a better understanding of how to easily manage files. In our case, the full path will become `/home/hadoopadmin/dockerImage`:

```
$ mkdir dockerImage
```

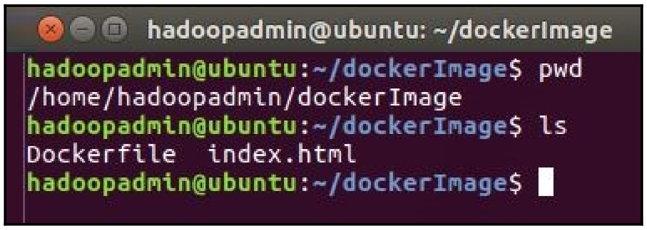
The next step is to create an HTML page. Open a text editor of your choice and write the following code in it, and then save it by the name `index.html` in the same folder as `dockerImage`:

```
<html>
<head>
  <title>Big Data Architect's Handbook</title>
</head>
<body>
  <h1>Welcome to Docker Container</h1>
  example from Big Data Architect's Handbook
</body>
</html>
```

The next step is to create a `Dockerfile`. This is a configuration file that builds a Docker image. It contains all the relevant information from the base operating system about different commands. These commands can be copied into a Docker image or be used to perform any configuration within the base operating system. Open a text editor of your choice and write down the following code. Save it by the name of `Dockerfile` within the same folder `dockerImage`:

```
FROM nginx:alpine

COPY ./index.html /usr/share/nginx/html
```

The preceding code contains two lines. The first line contains the command `FROM`, which tells Docker what the base image of the container will be. In our case, it is telling Docker to pull an NGINX web server image with an Alpine Linux base kernel. The second line tells Docker, once it has finished setting up the base system, to copy the `index.html` file from the current directory into the NGINX public directory, which is used to serve static pages. The following screenshot illustrates the directory structure and the files it contains:

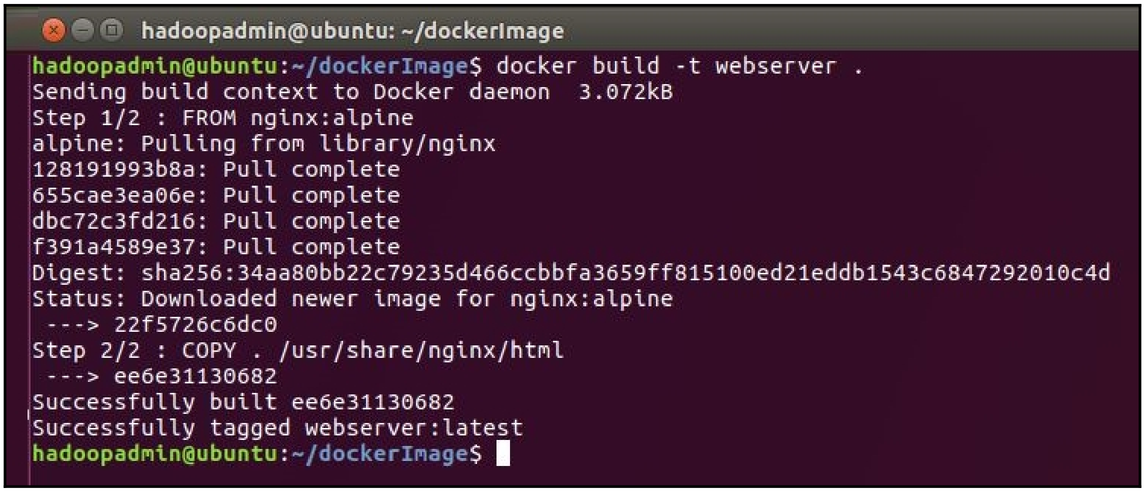


```
hadoopadmin@ubuntu:~/dockerImage$ pwd
/home/hadoopadmin/dockerImage
hadoopadmin@ubuntu:~/dockerImage$ ls
Dockerfile  index.html
hadoopadmin@ubuntu:~/dockerImage$
```

Docker build directory structure

Now, let's issue the following `build` command to create our first Docker image. The screenshot after the command displays its output:

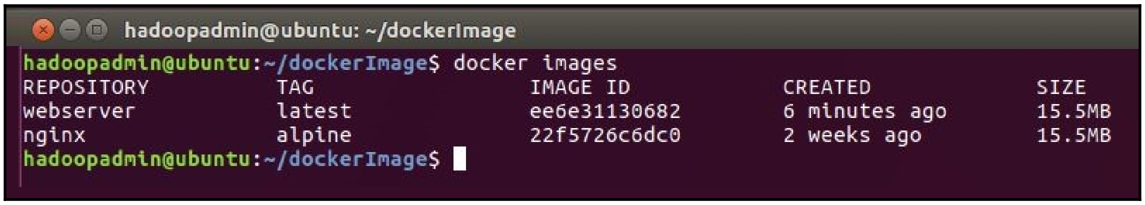
```
$ docker build -t webserver .
```



```
hadoopadmin@ubuntu: ~/dockerImage
hadoopadmin@ubuntu:~/dockerImage$ docker build -t webserver .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:alpine
alpine: Pulling from library/nginx
128191993b8a: Pull complete
655cae3ea06e: Pull complete
dbc72c3fd216: Pull complete
f391a4589e37: Pull complete
Digest: sha256:34aa80bb22c79235d466ccbbfa3659ff815100ed21eddb1543c6847292010c4d
Status: Downloaded newer image for nginx:alpine
 ---> 22f5726c6dc0
Step 2/2 : COPY . /usr/share/nginx/html
 ---> ee6e31130682
Successfully built ee6e31130682
Successfully tagged webserver:latest
hadoopadmin@ubuntu:~/dockerImage$
```

Build command output

Now, if you issue a list of Docker images, it will display two images. One is the base image downloaded from the Docker hub and the second one is a result of our `build` command. The following screenshot displays the output of the `docker images` command as verification:



```
hadoopadmin@ubuntu: ~/dockerImage
hadoopadmin@ubuntu:~/dockerImage$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
webserver           latest              ee6e31130682        6 minutes ago       15.5MB
nginx               alpine              22f5726c6dc0        2 weeks ago         15.5MB
hadoopadmin@ubuntu:~/dockerImage$
```

List of Docker images in the local repository

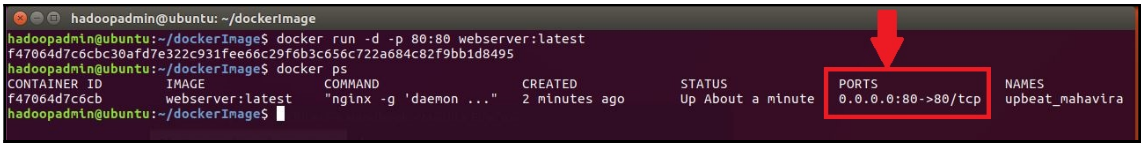
You can explore the full list of Docker images by visiting `https://hub.Docker.com/explore/`.

Running and verifying Docker images

We have successfully built our first Docker image, which runs an NGINX web server and hosts our static web page in a container. It's now time to test it. Issue the following command to run the Docker container:

```
$ docker run -d -p 80:80 webserver:latest
```

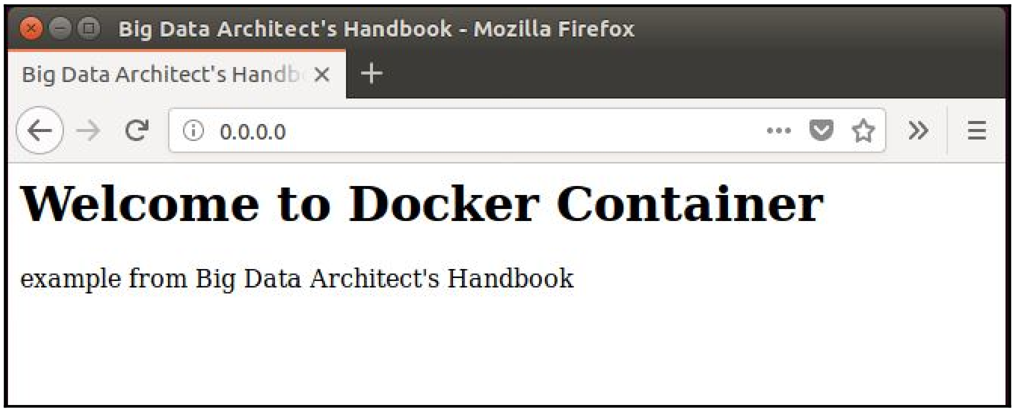
Once you have executed this command, it will display a process ID if it has successfully started the container. Issue the `docker ps` command to verify that the container has successfully started and that port `80` has been bound to this container, as shown in the following screenshot:



```
hadoopadmin@ubuntu: ~/dockerImage
hadoopadmin@ubuntu:~/dockerImage$ docker run -d -p 80:80 webserver:latest
f47064d7c6cbc30afd7e322c931fee66c29f6b3c656c722a684c82f9bb1d8495
hadoopadmin@ubuntu:~/dockerImage$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                NAMES
f47064d7c6cb        webserver:latest    "nginx -g 'daemon ..."   2 minutes ago       Up About a minute   0.0.0.0:80->80/tcp   upbeat_mahavira
hadoopadmin@ubuntu:~/dockerImage$
```

Start container and list running containers

It's now time to open a web browser and point it to the IP address mentioned in the `ps` command, as shown in the following screenshot:


Big Data Architect's Handbook - Mozilla Firefox
Big Data Architect's Handb
0.0.0.0

Welcome to Docker Container

example from Big Data Architect's Handbook

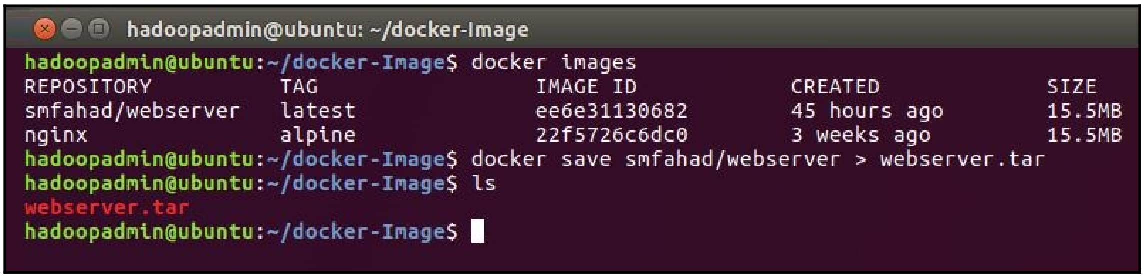
Opening a hosted website running inside a container

If you got the desired output, as shown in the preceding screenshot, it means that you have successfully built a Docker image and deployed it as a container. This container setup will work on a single node setup. We will explore what to do if you have a multi-node cluster environment shortly, but first let's learn how to export and import Docker images to and from your local filesystem.

Importing and exporting Docker images

Now, you have built your Docker image and tested it successfully. In the real world, you may be required to give this image to your testing or production team. In order to export your Docker image, execute the following command. The screenshot after the command displays its output:

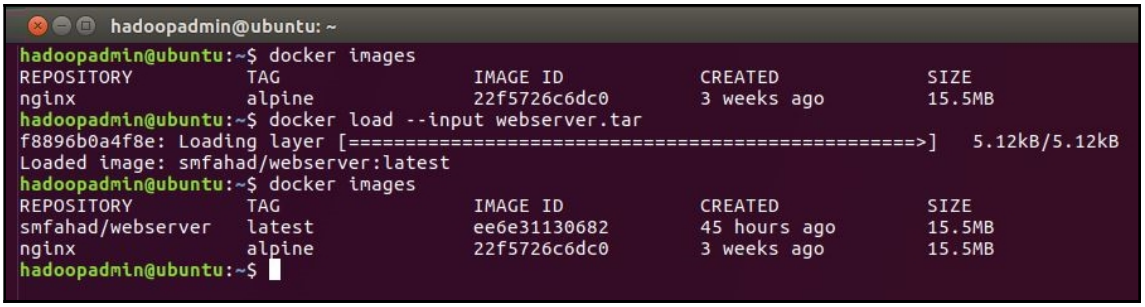
```
$ docker save smfahad/webserver > webserver.tar
```



```
hadoopadmin@ubuntu: ~/docker-Image
hadoopadmin@ubuntu:~/docker-Image$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
smfahad/webserver   latest              ee6e31130682        45 hours ago        15.5MB
nginx               alpine              22f5726c6dc0        3 weeks ago         15.5MB
hadoopadmin@ubuntu:~/docker-Image$ docker save smfahad/webserver > webserver.tar
hadoopadmin@ubuntu:~/docker-Image$ ls
webserver.tar
hadoopadmin@ubuntu:~/docker-Image$
```

Now, that you have the Docker image file, let's suppose you are now required to load it into your test or production environment. To do this, execute the following command. The following screenshot displays the output of this command:

```
$ docker load --input webserver.tar
```



```
hadoopadmin@ubuntu: ~
hadoopadmin@ubuntu:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               alpine              22f5726c6dc0        3 weeks ago         15.5MB
hadoopadmin@ubuntu:~$ docker load --input webserver.tar
f8896b0a4f8e: Loading layer [==================================================>]   5.12kB/5.12kB
Loaded image: smfahad/webserver:latest
hadoopadmin@ubuntu:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
smfahad/webserver   latest              ee6e31130682        45 hours ago        15.5MB
nginx               alpine              22f5726c6dc0        3 weeks ago         15.5MB
hadoopadmin@ubuntu:~$
```

Loading Docker images from a local file

Now that we've learned how to import and export Docker images, let's get back to the cluster environment setup. If you recall or revisit an earlier screenshot 4, which shows us detailed Docker information, you will notice a parameter named **Swarm**. You may be wondering what it is and why it is in an inactive state. Let's find out.

Docker Swarm

If you look the word up in a dictionary, 'swarm' will mean a number of things. Docker Swarm, however, is basically a cluster of machines, with each one running a Docker daemon. The primary purpose is to provide a scalable platform for a containerized application that can run in a clustered environment. In earlier versions of Docker, Swarm was available as a separate product, but from version 1.12 it no longer requires an additional application.

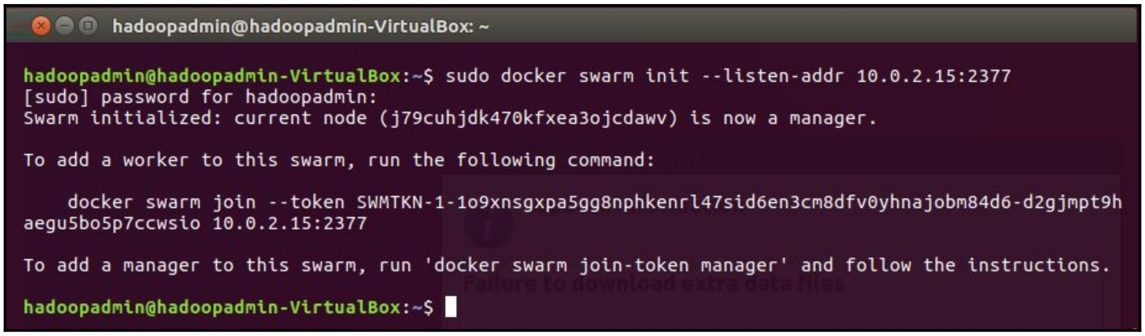
Docker Swarm works using the manager-worker concept, where one of the nodes becomes the manager and manages the other worker nodes. The prerequisites of running Docker Swarm is that all machines must be running Docker v1.12 or higher and must be available in the same subnet. Docker masters and slaves communicate with each other using ports. The default ports are `2377` for cluster management, `7946` for heartbeat and discovery, and `4789` for overlay network traffic. Make sure that these ports are not blocked by any of your machines with a firewall to ensure continued Docker Swarm communication.

Setting up Docker Swarm

Setting up Swarm is quite simple. First, let's assume that we have six machines. Their IP addresses are from `10.0.2.15` to `10.0.2.20`, which means they belong to the same subnet. Install Docker version 1.12 or higher in all of these machines, as per the steps discussed in the Docker Installation section of this chapter. After successful installation and verification on these machines, it's time to do Swarm setup.

The first step is to decide which machine you want to make manager. In our case, we will make `10.0.2.15` our Swarm manager. In order to do this, execute the following command. The following screenshot illustrates the output of the following command:

```
$ sudo docker swarm init --listen-addr 10.0.2.15:2377
```

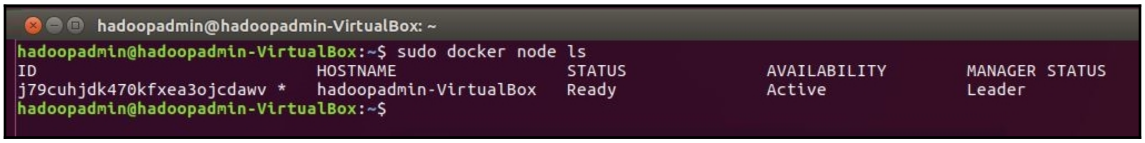


```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker swarm init --listen-addr 10.0.2.15:2377
[sudo] password for hadoopadmin:
Swarm initialized: current node (j79cuhjdk470kfxea3ojcdawv) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1o9xnsgxpa5gg8nphkenrl47sid6en3cm8dfv0yhnajobm84d6-d2gjmpt9h
aegu5bo5p7ccwsio 10.0.2.15:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

hadoopadmin@hadoopadmin-VirtualBox:~$
```

Initializing Docker Swarm

The output, as you can see, is displaying a message that says the current node is now a manager. Another way to list the nodes available in the Swarm along with their status (whether they are a worker or a manager) is to execute the following command. The screenshot following the command displays the output of the following command:

```
$ sudo docker node ls
```



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker node ls
ID                           HOSTNAME                STATUS              AVAILABILITY        MANAGER STATUS
j79cuhjdk470kfxea3ojcdawv *   hadoopadmin-VirtualBox  Ready               Active              Leader
hadoopadmin@hadoopadmin-VirtualBox:~$
```

List of available nodes in Docker Swarm

Now, to add a worker node in Swarm, log in to the machine that you want as your worker. In our case, it will be `10.0.2.16`. Execute the following command from its Terminal:

```
$ docker swarm join 10.0.2.15:2377
```

Once you have executed the command, it will display a successful message saying that the current node has joined Swarm as a worker. The important thing to notice here is that in this command, you have to pass the IP address of the manager node to join. You can do the same with all of the remaining machines. Log in to them and execute the preceding command from their Terminals.

You can verify the list of joined nodes by executing the nodes `ls` command, as shown earlier, from the manager's Terminal. This will list all the nodes along with their name, status, availability, and managerial status.

Creating service containers

A service is referred to as a **container** in Docker and is managed by Swarm itself. In other words, Swarm runs the desired container, but I am not going to manage the individual instances of that container. We can also define the replication factor, which distributes instances of the desired service among different nodes in the Swarm.

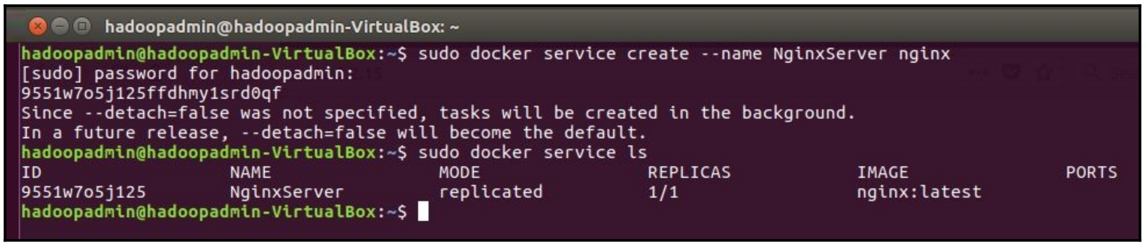
For demonstration purposes, we will be running an NGINX image in the container. NGINX is an open source web server. We will not be discussing NGINX or any of its functionalities in this section; we will just be using its image to understand the service and replication concept in Docker Swarm. First, let's create a service. Execute the following command in the manager node Terminal:

```
$ docker service create --name NginxServer nginx
```

To verify that the services have been created and are running in the background, execute the following command:

```
$ sudo docker service ls
```

The following screenshot displays the output of both the `service` commands:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service create --name NginxServer nginx
[sudo] password for hadoopadmin:
9551w7o5j125ffdhmy1srd0qf
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
9551w7o5j125        NginxServer         replicated          1/1                 nginx:latest
hadoopadmin@hadoopadmin-VirtualBox:~$
```

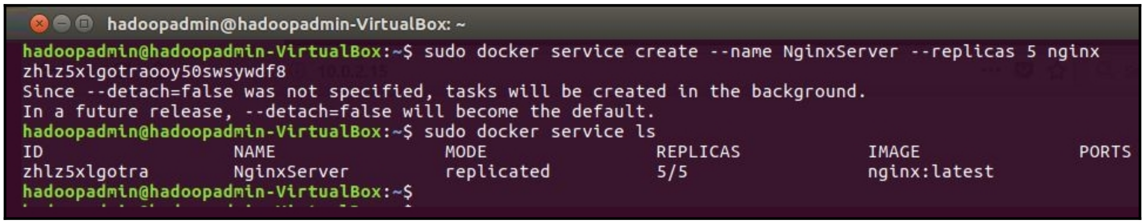
Creating and verifying a Swarm service

Replicating containers

As you can see, the service `ls` command lists the service that we have just created with a replication factor of 1/1. Let's suppose that we want to create a service with a replication factor of `5`. Execute the following command to fulfil that requirement:

```
$ docker service create --name NginxServer --replicas 5 nginx
```

The preceding command will initiate five instances of the container so that if any of the containers fail, another will be there to service the request or whatever task it was assigned to perform. The following screenshot displays the output of this command:

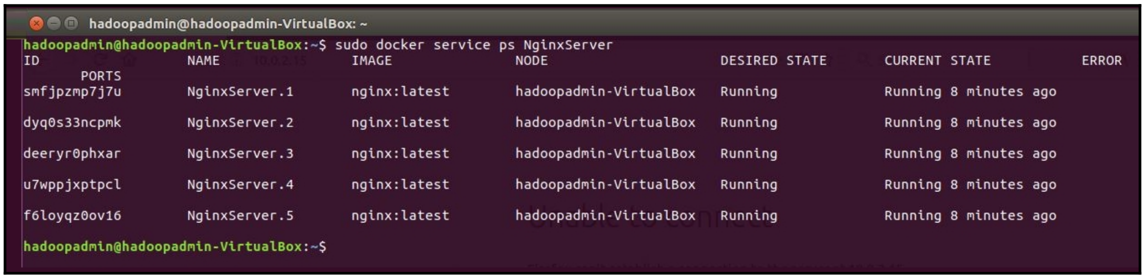


```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service create --name NginxServer --replicas 5 nginx
zhlz5xlgotraooy50swsywdf8
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zhlz5xlgotra        NginxServer         replicated          5/5                 nginx:latest
hadoopadmin@hadoopadmin-VirtualBox:~$
```

ervice creation with replication

To list the details regarding each replicated service, execute the following command. The following screenshot displays the output of this command:

```
// Replace NginxServer with the desired service name
$ sudo docker service ps NginxServer
```



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service ps NginxServer
ID                  NAME                IMAGE               NODE                       DESIRED STATE       CURRENT STATE           ERROR
         PORTS
smfjpzmp7j7u        NginxServer.1       nginx:latest        hadoopadmin-VirtualBox     Running             Running 8 minutes ago

dyq0s33ncpmk        NginxServer.2       nginx:latest        hadoopadmin-VirtualBox     Running             Running 8 minutes ago

deeryr0phxar        NginxServer.3       nginx:latest        hadoopadmin-VirtualBox     Running             Running 8 minutes ago

u7wppjxptpcl        NginxServer.4       nginx:latest        hadoopadmin-VirtualBox     Running             Running 8 minutes ago

f6loyqz0ov16        NginxServer.5       nginx:latest        hadoopadmin-VirtualBox     Running             Running 8 minutes ago

hadoopadmin@hadoopadmin-VirtualBox:~$
```

Details of each replication of a desired service

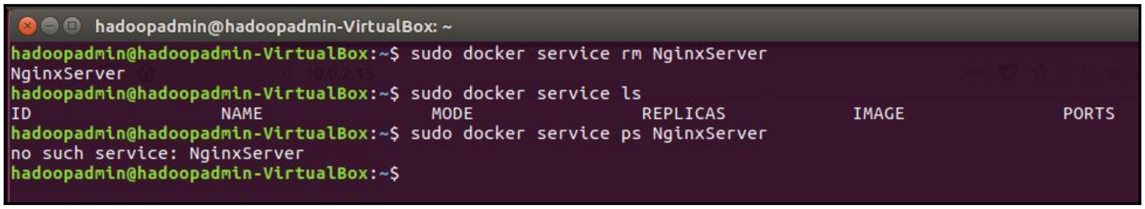
This detail gives you a lot of useful information, such as what a service's current status is, in which nodes the replicated containers are running, whether it has encountered any errors, and so on.

Removing container services

Once your desired task is complete, or if you want to stop or delete a service, execute the following command:

```
$ sudo docker service rm NginxServer
```

The following screenshot illustrates the output of this command, as well as service `ls` and the detailed task command for verification purposes:



```
hadoopadmin@hadoopadmin-VirtualBox: ~
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service rm NginxServer
NginxServer
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
hadoopadmin@hadoopadmin-VirtualBox:~$ sudo docker service ps NginxServer
no such service: NginxServer
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Removing a service

We have looked at containerization using Docker in detail, including environment and cluster setup. We have also gone through different flavors of Docker Swarm. Now, let's look into another tool available on the market that automates container deployment, management, and scaling: Kubernetes.

Kubernetes

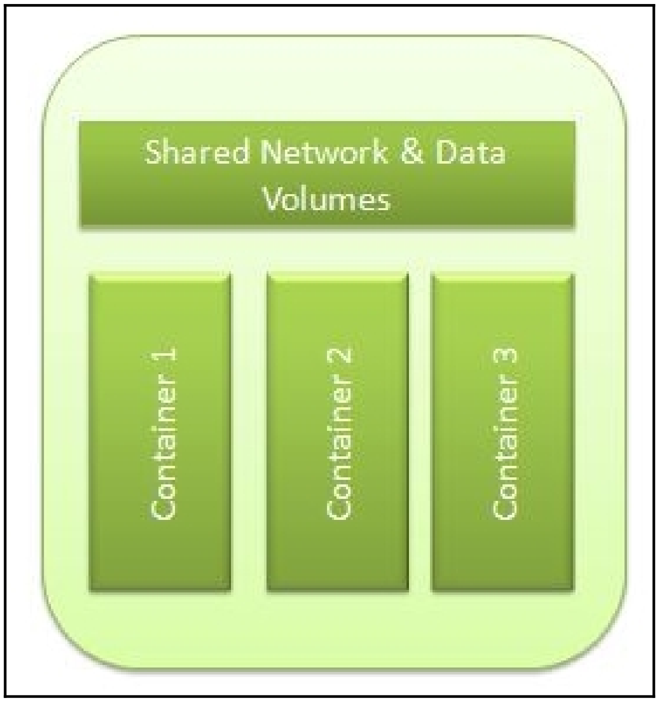
Kubernetes is an open source tool for orchestrating containers in a cluster. It supports different container technologies including Docker and rkt. It was initially designed by Google and was based on its internal cluster management project **Borg** and was named **Project Seven**. It was then passed on to Cloud Native Computing Foundation as an initial technology in July 2015.

Key components

There are some key components and terminologies you will encounter while using Kubernetes. These are as follows.

Pods

A **pod** is one or more of a group of containers that are tightly coupled to each other. The container can be based on any framework, such as Docker, rkt, and so on. It the smallest deployable unit. It also has shared network and data volumes. The following screenshot illustrates the internal structure of a pod:



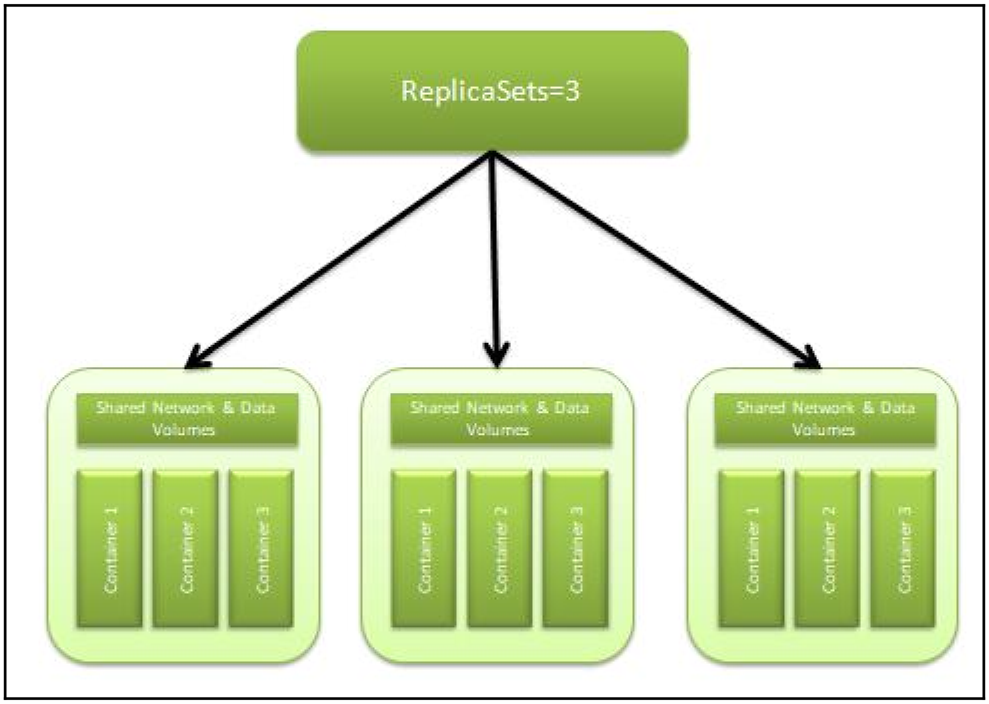
Internal structure of a pod

A pod has its own routable IP address, but it is not a requirement that it always has the same address. For example, if a failure occurs and a pod has to restart, it may restart in the same cluster node or in a different node, which may lead to a different IP address.

You cannot replicate a pod; it is a result of ReplicaSets, which we will discuss next.

ReplicaSets

The purpose of **ReplicaSets** is to create any number of copies of a pod. It is responsible for starting and stopping a pod. It also regularly checks the status of a pod. If any pod faces any issue or failure, it gets restarted. The following figure illustrates ReplicaSets:



The structure of a ReplicaSet

We cannot create ReplicaSets manually as it's part of the deployment, which we will discuss next.

Deployments

A **deployment** is a configuration file that contains all the declarations of your application. It includes declarations for environment variables, container images, data volumes, and much more. It also contains information related to ReplicaSets. Once you execute a deployment, it creates ReplicaSets, which in turn creates pods.

When you update your application, it creates a separate ReplicaSets as a result of your new deployment. It kills a pod from the old version of your application and creates a new pod as per the new version of the application - and the cycle goes on. If anything goes wrong as a result of a new deployment, it will automatically roll back to the previous version of your application.

PetSets

A **pet** is a pod that has a state. It means that the data volume that is dynamically created, as required by a container, will never be deleted automatically. It will again reassociate to the same data volume with which it was first associated upon restart. A collection of these pets is referred to as a PetSet.

Now that we have gone through some of the key components of Kubernetes, let's get on with something practical.

Installation

Kubernetes requires a cluster of computers to set it up; you'll need multiple computers or VMs in order to set up Kubernetes. Thanks to Minikube, however, you can get started with just a single node. We will also look at the installation of Minikube, so as to get a flavor of Kubernetes.

The first component that is required on the road to installation is `kubectl`. This is a command line interface that interacts with Kubernetes. Execute the following command to download it on your computer:

```
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl
-s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/l
inux/amd64/kubectl
```

This command will download the latest stable version of `kubectl` to your computer. To make it executable, run the following command:

```
$ chmod +x ./kubectl
```

Now, move the executable of `kubectl` to the PATH of your binaries so that the `kubectl` command will be accessible from your Terminal in any location:

```
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

Now that we have set up `kubectl`, let's install Minikube. Execute the following command to download it:

```
$ curl -Lo minikube
https://storage.googleapis.com/minikube/releases/v0.24.1/minikube-linux-amd
64
```

Execute the following command to make it executable:

```
$ chmod +x minikube
```

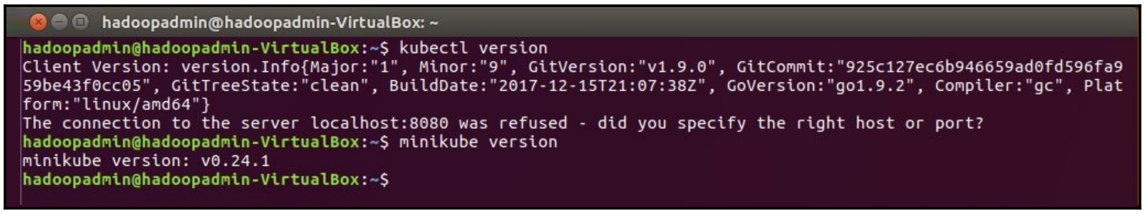
Now, in a similar fashion to `kubectl`, move the `minikube` executable to the PATH of your binaries by executing the following command:

```
$ sudo mv Minikube /usr/local/bin/
```

In order to verify that `kubectl` and `Minikube` installed successfully, execute the following command:

```
$ kubectl version
$ minikube version
```

The following screenshot displays the output of the `verification` commands:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"9", GitVersion:"v1.9.0", GitCommit:"925c127ec6b946659ad0fd596fa9
59be43f0cc05", GitTreeState:"clean", BuildDate:"2017-12-15T21:07:38Z", GoVersion:"go1.9.2", Compiler:"gc", Plat
form:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
hadoopadmin@hadoopadmin-VirtualBox:~$ minikube version
minikube version: v0.24.1
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Kubectl and Minikube installation verification

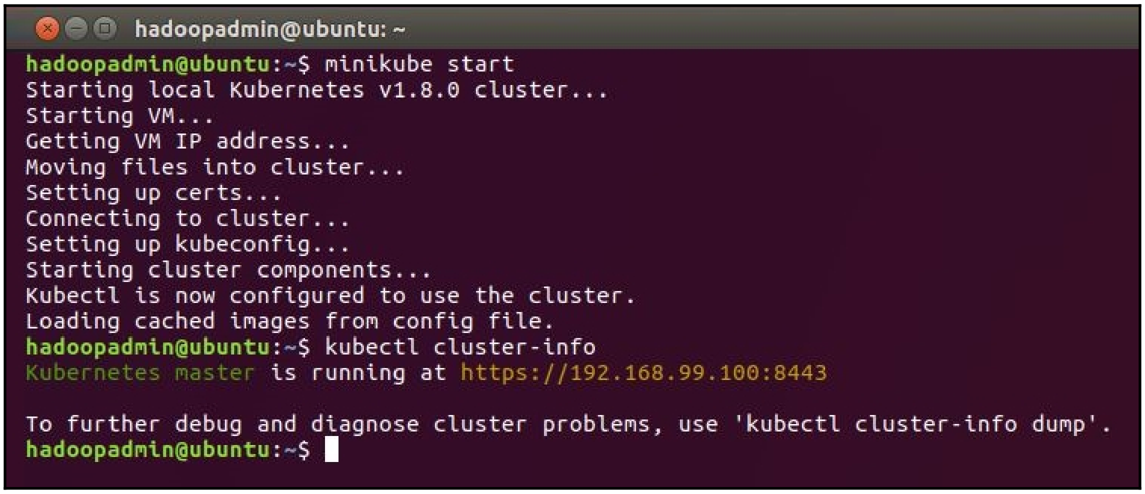
If you see an output similar to the preceding screenshot, it means that both installations went as expected. If the `verification` command displays some other message, perform the installation again.

The next step is to start Minikube. Before that, however, you must install any of the hypervisors. In our case, we will install VirtualBox. Download VirtualBox from their website, `https://www.virtualbox.org/wiki/Downloads`, based on your operating system. Once the download is complete, simply double click on the package to install it.

Now that we have successfully installed VirtualBox, it's time to start Minikube. For this purpose, execute the following command:

```
$ minikube start
```

Once you execute the command, it will start downloading multiple files that are almost 280 MB in size. The following screenshot displays the output you will get, along with the `verification` command, once you start Minikube:



```
hadoopadmin@ubuntu: ~
hadoopadmin@ubuntu:~$ minikube start
Starting local Kubernetes v1.8.0 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
Loading cached images from config file.
hadoopadmin@ubuntu:~$ kubectl cluster-info
Kubernetes master is running at https://192.168.99.100:8443

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
hadoopadmin@ubuntu:~$
```

Minikube startup with verification

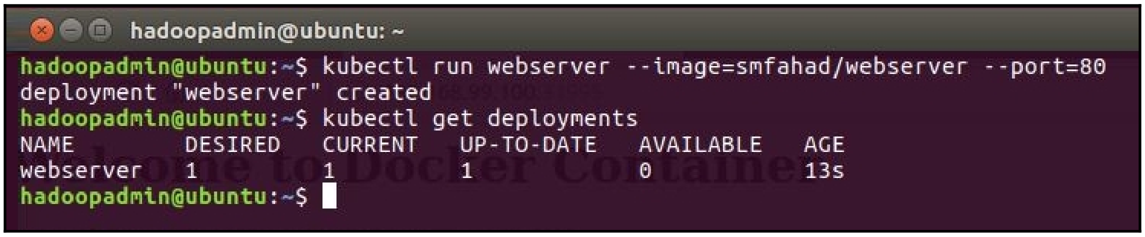
As you can see, the cluster has been started successfully. You can access the Kubernetes master node from the IP address `192.168.99.100`, although it may be different in your case. In the following section, we will learn how to deploy a container in your Kubernetes cluster.

Deployment

We will use the same Docker image that we created in the *Docker image* section of this chapter, and will deploy and run it in our Kubernetes cluster. In order to do this, execute the following command:

```
// Use this command to deploy docker image
$ kubectl run webserver --image=webserver --port=80

// Verification command
$ kubectl get deployments
```

The following screenshot displays the execution result and the output of the `verification` command:

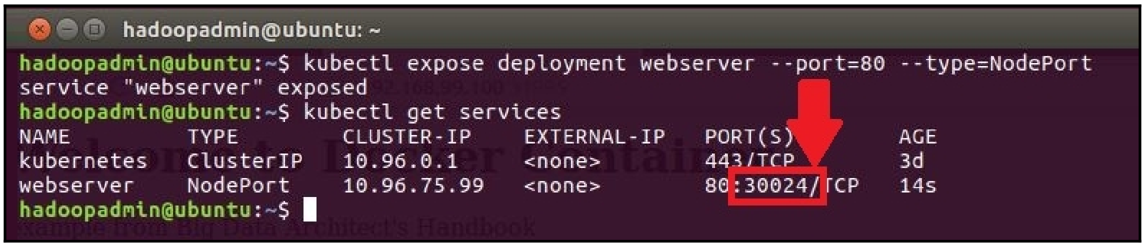


```
hadoopadmin@ubuntu: ~
hadoopadmin@ubuntu:~$ kubectl run webserver --image=smfahad/webserver --port=80
deployment "webserver" created
hadoopadmin@ubuntu:~$ kubectl get deployments
NAME        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webserver   1         1         1            0           13s
hadoopadmin@ubuntu:~$
```

Docker Image deployment and verification

The next step is to expose the deployed image so that you can access the container using port `80`, which is best suited for `http` requests. Execute the following command in order to do so. The following screenshot displays the output of this, as well as the `verification` command:

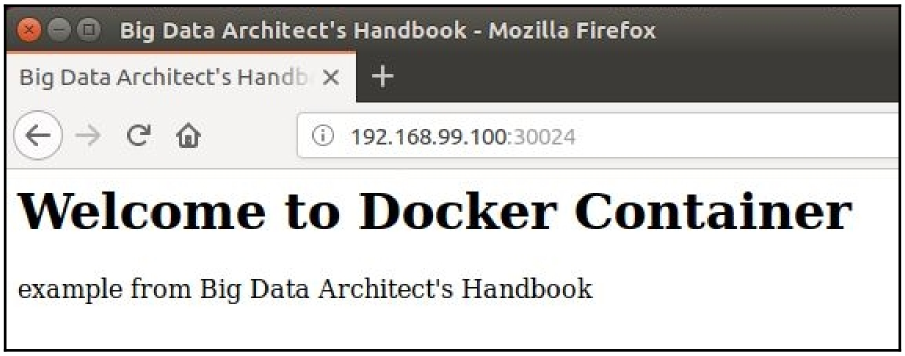
```
// Expose the deployed service for external use
$ kubectl expose deployment webserver --port=80 --type=NodePort

//Verification command
$ kubectl get services
```



```
hadoopadmin@ubuntu: ~
hadoopadmin@ubuntu:~$ kubectl expose deployment webserver --port=80 --type=NodePort
service "webserver" exposed
hadoopadmin@ubuntu:~$ kubectl get services
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP        3d
webserver    NodePort    10.96.75.99   <none>        80:30024/TCP   14s
hadoopadmin@ubuntu:~$
```

Results of the expose and verification commands

Now, open your browser and point the address bar to the IP address of your Kubernetes master node, along with the port number highlighted in the preceding screenshot. In our case, it will become `http://192.168.99.100:30024`. The following screenshot displays the output of the browser once you have pointed it to the mentioned URL:


Big Data Architect's Handbook - Mozilla Firefox
Big Data Architect's Handb
192.168.99.100:30024

Welcome to Docker Container

example from Big Data Architect's Handbook

Webserver output running in a container

Kubernetes Dashboard

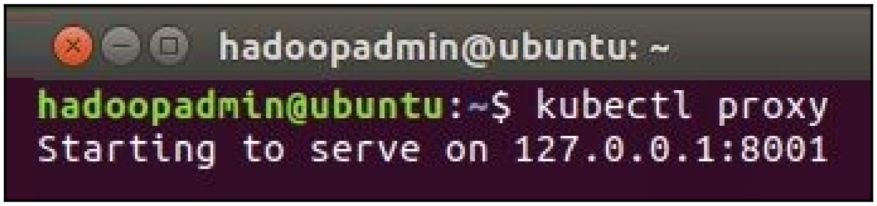
Kubernetes provides a web UI-based dashboard interface. Here, you can perform different actions, for example, deploying a container, scaling, and so on. It also provides an overview of resource utilization, deployments, pods, and more. Follow these steps to install the Kubernetes Dashboard and start exploring:

1. Execute the following command to install it:

```
$ kubectl create -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/src/d
eploy/recommended/kubernetes-dashboard.yaml
```

2. Once the execution of the installation command is complete, you can execute the following command to start serving the dashboard. The following screenshot displays the output of this command:

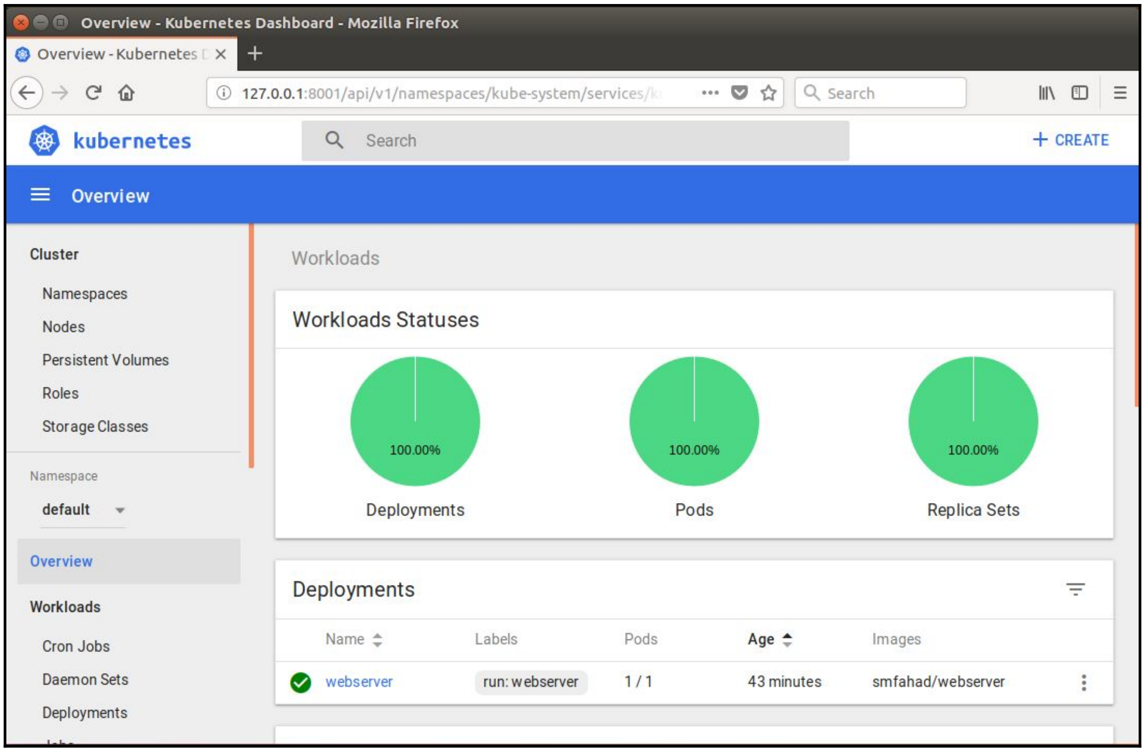
```
$ kubectl proxy
```



```
hadoopadmin@ubuntu:~$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Kubernetes Dashboard

3. Point your browser to `http://127.0.0.1:8001/ui/`. It will display an output similar to the one shown in the following screenshot:



Name	Labels	Pods	Age	Images
webserver	run: webserver	1 / 1	43 minutes	smfahad/webserver

Kubernetes Dashboard UI

If you can see a website similar to the preceding screenshot, it means that the Kubernetes Dashboard has been successfully installed and you can start using and exploring it.

Summary

In this chapter, we learned about virtualization and different types of hypervisor. Then, we learned about containerization and its benefits. Nowadays, containers are a hot topic in the virtualization world, and the primary framework leading the way is Docker. In this chapter, we learned about Docker and its workflow. Moving from the theoretical to the practical, we then proceeded with its installation, along with learning some basic manipulation commands that deal with Docker. We then learned how to create a Docker image and how to deploy a sample static website in a container, as well as how to export and import Docker images if and when required. We also learned about Docker Swarm, which is used to scale Docker containers to a multi-node cluster.

Kubernetes is a very prominent component in cluster management for containers. We learned about its different components before installing Minikube, which is used to explore the features of Kubernetes' cluster management in a single node environment. We then learned how to deploy a Docker container in a Kubernetes cluster and verify its working status. Lastly, we installed the Kubernetes Dashboard, which is used to perform different command-line actions in a web-based environment along with resource allocation overviews and details.

In the next chapter, we will explore different design considerations for big data setups over cloud infrastructure.

7
Network Infrastructure

The network is the backbone of any infrastructure. If you have high-end machines at your disposal but the network is not well-designed or not present at all, then their resources will be limited just to their own, or we will not be able to utilize their full potential.

In this chapter, we will be going through the following different network concepts and terminologies to give you an idea about the network industry:

- Network
- LAN, MAN, WAN
- Wired, WiFi, VPN
- Network visualization
- Gephi – a network visualization tool

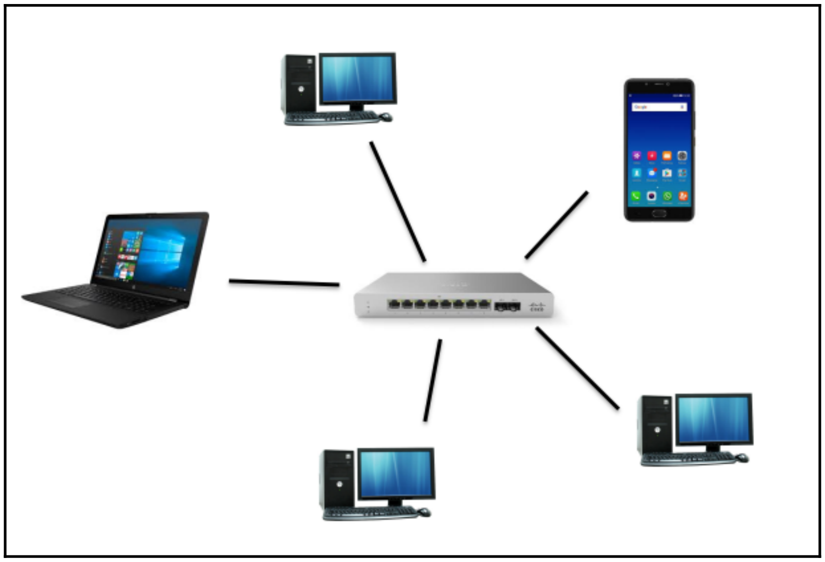
Let's start by understanding what a network is and what common types of networks we use.

Network

When we talk about a network in general, we refer to when any two objects are able to communicate with each other using any means possible. Having this in mind, we will now talk about computer networks; in this type of network, two computers are connected in a way that they can share their hardware as well as software resources. Now, the way they connect to each other under different environmental setups depends on different configurations, which we will explore in the subsequent sections.

Local area networks

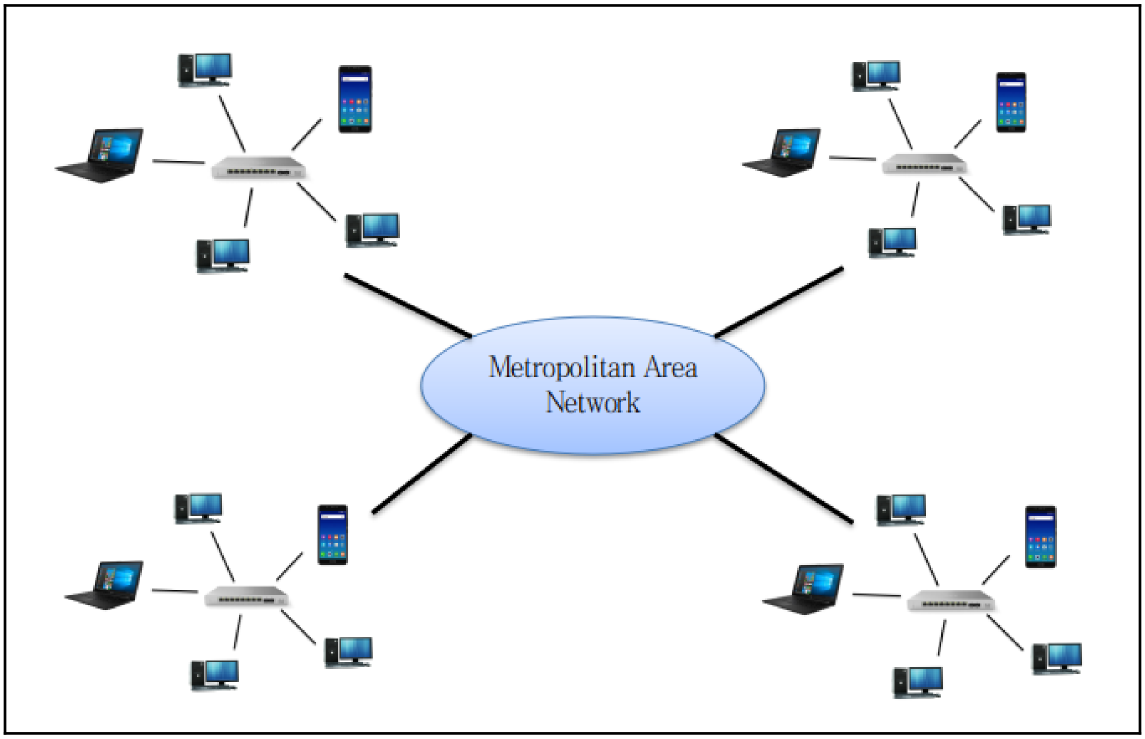
Local area networks are also referred to as **LANs**. This type of network is usually configured within a house or in an office, a school building, and so on. The devices are mostly interconnected with the help of a device called a router or a switch. Routers support both wired or wireless connections whereas switches only support connectivity via wired. The following diagram illustrates the architecture of a local area network:



Architecture of a local area network

Metropolitan area networks

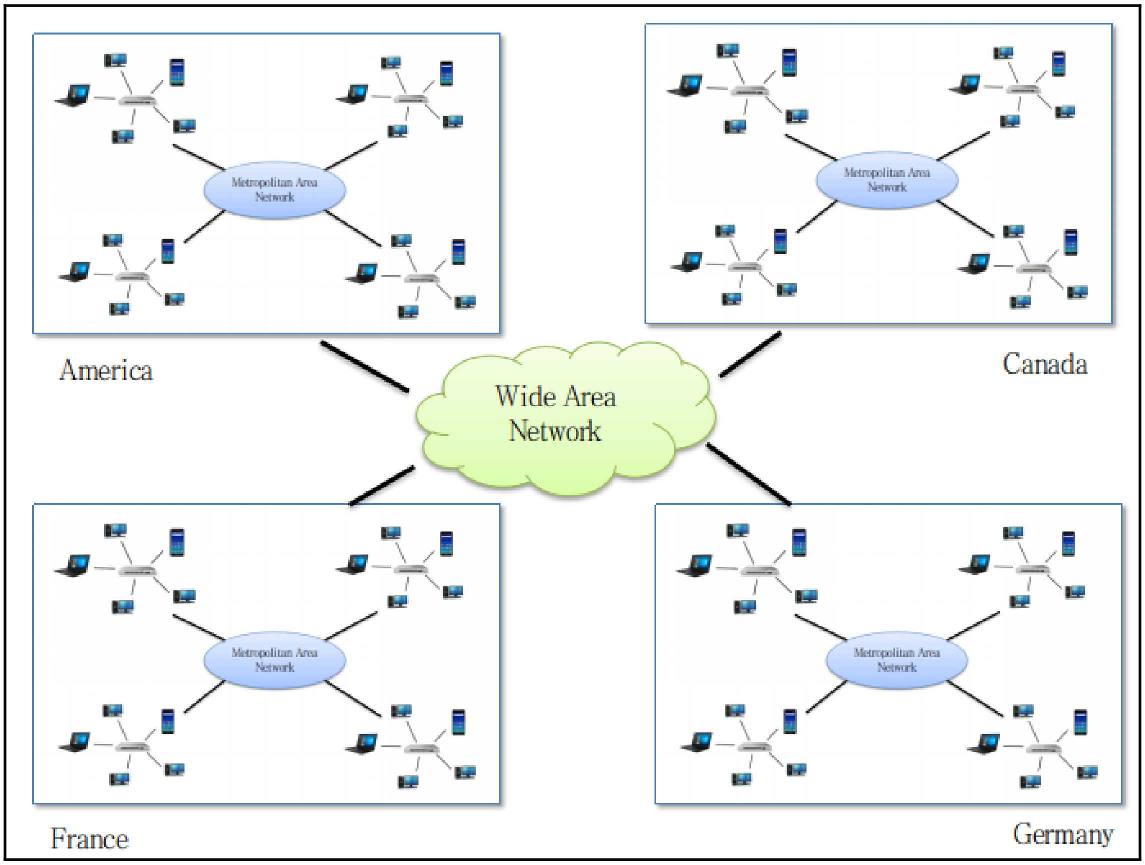
Metropolitan area networks are also referred to as **MANs**. This type of network is usually established to interconnect different geographical locations, such as two cities or remote areas. It is bigger then local area networks but smaller that wide area networks. In simpler terms, MAN is made by connected different large local area networks using point to point connection. The following diagram illustrates the architecture of a metropolitan area network:



Metropolitan area network

Wide area networks

Wide area networks are also referred to as **WANs**. These are spread over different geographical locations spread over large distances. For example, connectivity between two cities of different countries. This type of setup is normally established between different parties with the help of a contract. This type of contract is normally used to rent the services of the other party. In simpler words, a WAN is a collection of different metropolitan networks interconnected to each other. The following diagram illustrates the architecture of a wide area network:


Metropolitan Area Network
America
Metropolitan Area Network
Canada
Wide Area Network
Metropolitan Area Network
France
Metropolitan Area Network
Germany

Architecture of a wide area network

Now that we have established an understanding about different types of networks, let's develop an understanding of the different connectivity methods used to connect them.

Network connectivity

There are various ways to establish a network connection. The method of connectivity varies based on the network size and the devices it connects to. The connectivity can be different types of wired connections or different technologies used in wireless connection. We will discuss some in detail in the subsequent sections.

Wired

In this type of setup, computers are connected to each other by means of a hard wire, which is commonly known as an Ethernet cable. In the basic configuration of this type of setup, one end of the Ethernet cable will connect to a computer and the other end will connect to a router or a switch. All available computers connect to a switch or router in a similar fashion to form a network. This is a typical example of a local area network.

Wireless

With the increase in IoT and handheld devices, network setup can also be established in a wireless environment. Routers act as a base station and all the wireless devices act as a client. They follow the IEEE 802.11 standard to establish communication between them. There are different bands, which are normally referred to as
802.11, 802.11a, 802.11b, 802.11g, 802.11n, and the dual band channel, 802.11ac. There are different series within the 802.11ac band, such as 1200, 1500, 1700, and 3200. The main difference between them is the communication speed. The following chart lists the communication speeds of each band:

IEEE standard	**Wireless speed**
802.11	2 Mbps
802.11 b	11 Mbps
802.11 a	54 Mbps
802.11 g	54 Mbps with greater range
802.11 n	300 Mbps
802.11 ac	450 Mbps on 2.5 Ghz and 1300 Mpbs on 5 Ghz

Now that we have developed an understanding of the different types of network structure as well as the different types of connectivity between them, we will now move on to visualize a network with the help of an interactive tool.

Network visualization

Usually, in large network setup, it is difficult to visualize the interconnectivity of different nodes and the amount of resources they share between them. If the network size is small then the analysis is not a big issue, but if the network size is big, then it is difficult to make any sort of analysis on it. For this purpose, there are different open source and commercial tools available on the market. The tool that we will be exploring is called **Gephi**, and we will be exploring more about it in the following section.

Gephi

Gephi is a network visualization tool for analyzing a network in a graphical format. It is an open source application. It displays different types of graphs and charts for better understanding and analysis. It provides many options for customizing the working dataset and interacting with graphs and customizing their appearances to identify patterns. The following are some of the key features of Gephi:

- Real time visualization
- Layout algorithm
- Statistics and Metrics framework
- Network timeline
- Cryptography customization
- Dynamic filtering
- Data table and editing
- Import and export to/from different sources

Let's dive in more and start the installation process. Afterwards, we will perform the actual practical on it.

Installation

The first step is to download the required application. This can be done by visiting `https://gephi.org/users/download/`. At the time of writing this book, the current version of Gephi is 0.9.2. Once you visit the previous URL, it will display a button to download the Gephi application based on your operating system. In our case, it is Ubuntu, so it will display a button that says **Download Gephi for Linux**. Once you click on it, it will download the file `gephi-0.9.2-linux.tar.gz`. Copy this file into your home folder and execute the following command to extract it:

```
$ cd ~/
$ tar -zxvf gephi-0.9.2-linux.tar.gz
```

This command will extract the file containing the `gephi` package. That's it for the installation part. If you're doing it on a new system, then you will have to configure Java first.

Java installation

To install Java, use the following instructions:

1. Download JDK and JRE from `http://www.oracle.com/technetwork/java/javase/downloads/index.html`.
2. Copy the downloaded `jdk-8u144-linux-x64.tar.gz` file to the location where you want to install Java.
3. Extract and install Java by using the following command. Make sure to replace `8u144` with the Java version you have downloaded:

   ```
   $ tar zxvf jdk-8u144-linux-x64.tar.gz
   ```

4. The next step is to point the system to the extracted directory. In our case, files are extracted to the `/home/fahad/jdk1.8.0_144/` folder. Execute the following command to make this Java version the default JVM:

   ```
   $ sudo update-alternatives --install /usr/bin/java java
   /home/fahad/jdk1.8.0_144/bin/java 100
   $ sudo update-alternatives --install /usr/bin/javac javac
   /home/fahad/jdk1.8.0_144/bin/javac 100
   ```

5. To make sure Java is installed correctly, use the following command to check your Java version:

   ```
   $ java -version
   ```

6. Edit the `.bashrc` file using the following command. Make sure not to make any other changes in this file:

   ```
   $ sudo gedit ~/.bashrc
   ```

7. Here, we will configure the Java path. Copy the following code at the end of this file:

   ```
   export JAVA_HOME="/home/fahad/jdk1.8.0_144"
   export PATH=$PATH:$JAVA_HOME/bin
   ```

8. Press *Ctrl* + *O* and then *Ctrl* + *X* to exit the editor. Now, execute the following command to reload the environment variables:

   ```
   $ source ~/.bashrc
   ```

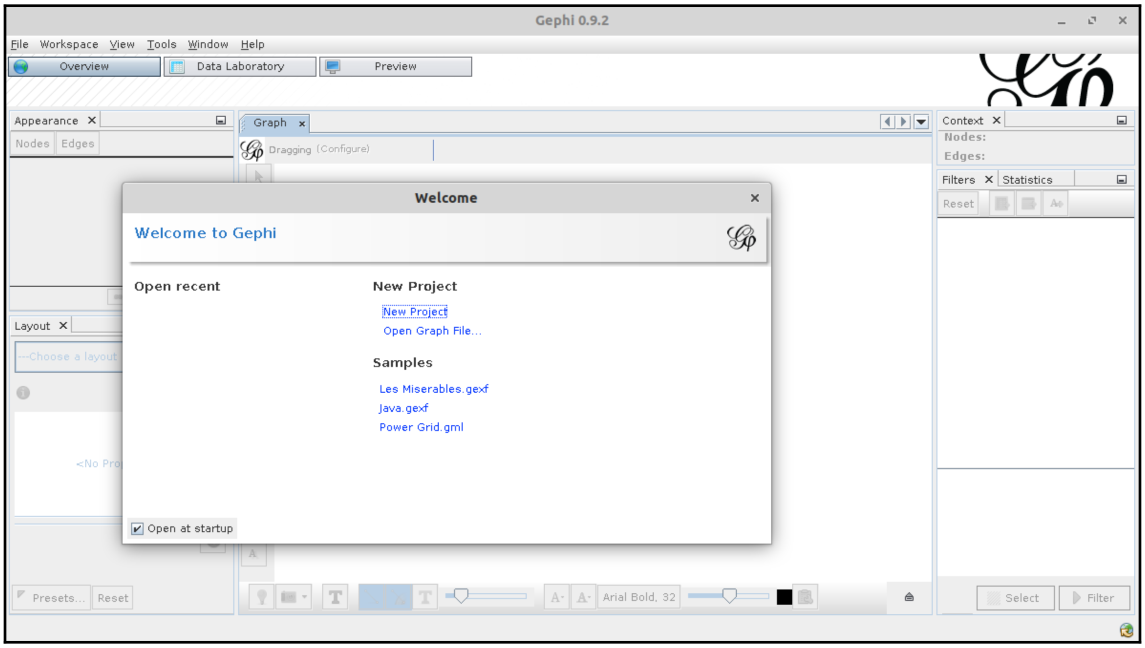
9. Now that you have successfully installed and configured Java, you can delete the downloaded file as it is no longer required.

First run

Now that we have configured Java and Gephi, we will now issue the following command in order to start the application:

```
$ cd ~/gephi-0.9.2/
$ sudo ./bin/gephi
```

Once you execute this command, it will open up the application, as shown in the following screenshot:



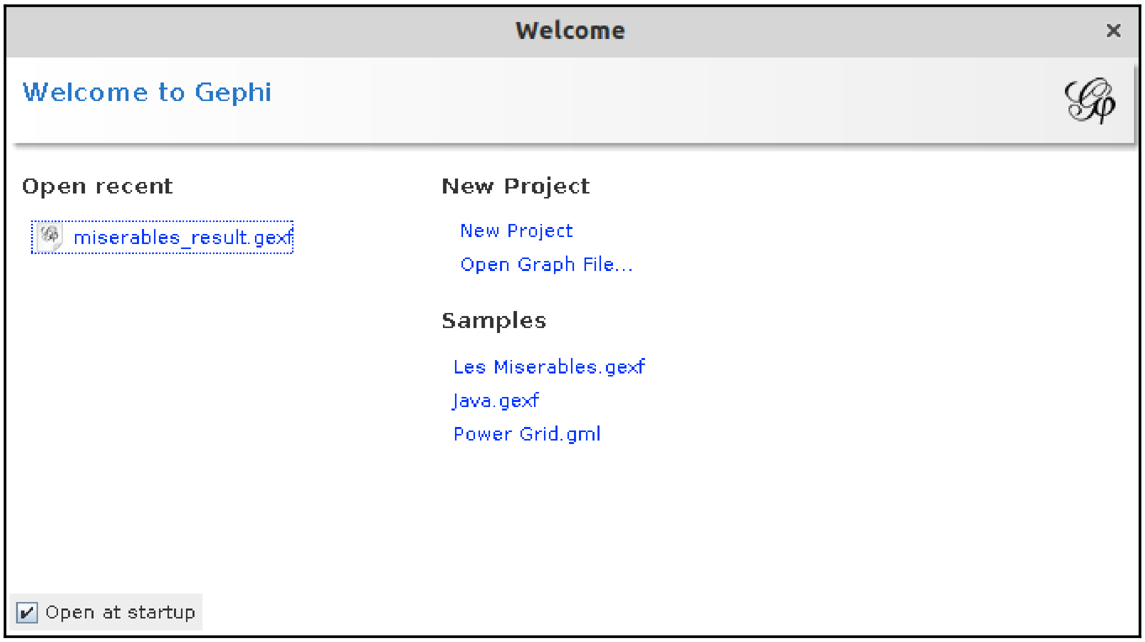
Gephi first run

Now we are all set and ready to simulate our first practical example.

Practical example

In this practical example, we are going to create a visual representation of a local area network in which two computers, two mobile phones, and a laptop are connected to a router. Let's start by firing up the Gephi application, as mentioned in the previous section. Follow this step by step guide to create this visual representation:

1. The first step is to click on the **New Project** link, as shown in the following screenshot:


Welcome

Welcome to Gephi

Open recent

miserables_result.gexf

New Project

New Project

Open Graph File...

Samples

Les Miserables.gexf

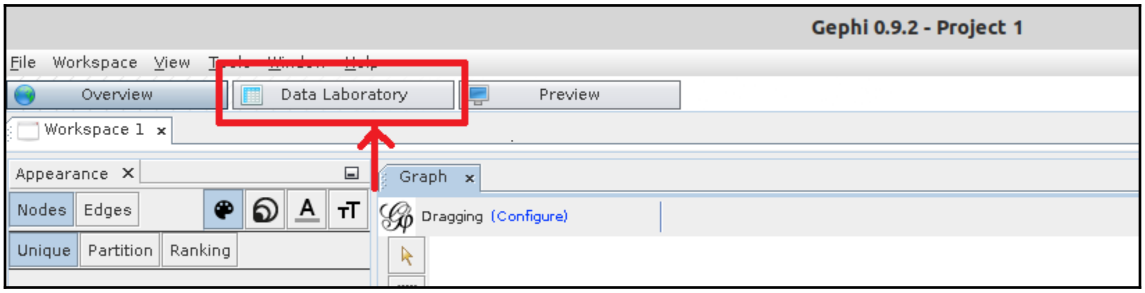
Java.gexf

Power Grid.gml

Open at startup

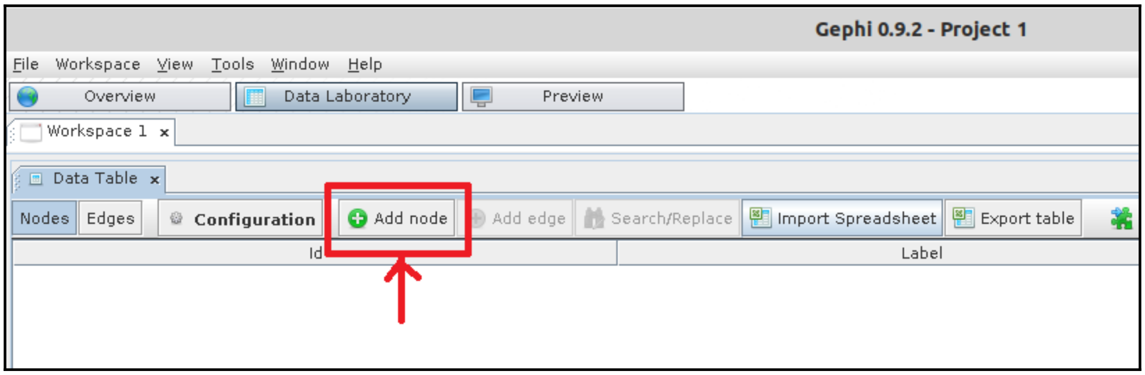
Starting a new project in Gephi

2. This will open up a new project for you. The next step is to switch to **Data Laboratory** mode to start inserting some nodes in your network. Do so by clicking on **Data Laboratory** on the top left side of the application, as shown in the following screenshot:



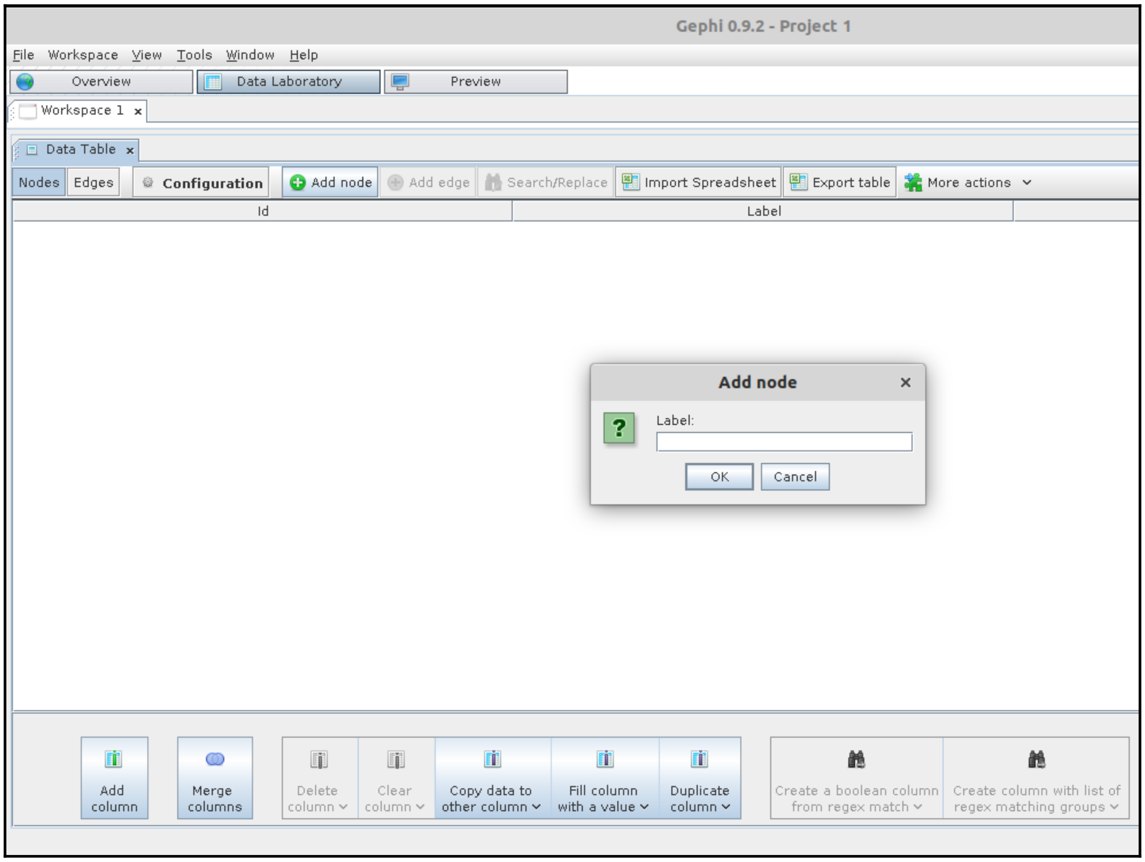
Switching to the Data Laboratory interface

3. Once you click on it, it will switch the interface to **Data Laboratory** mode. The next step is to click on **Add Node**, as shown in the following screenshot, to create a new node in your network:



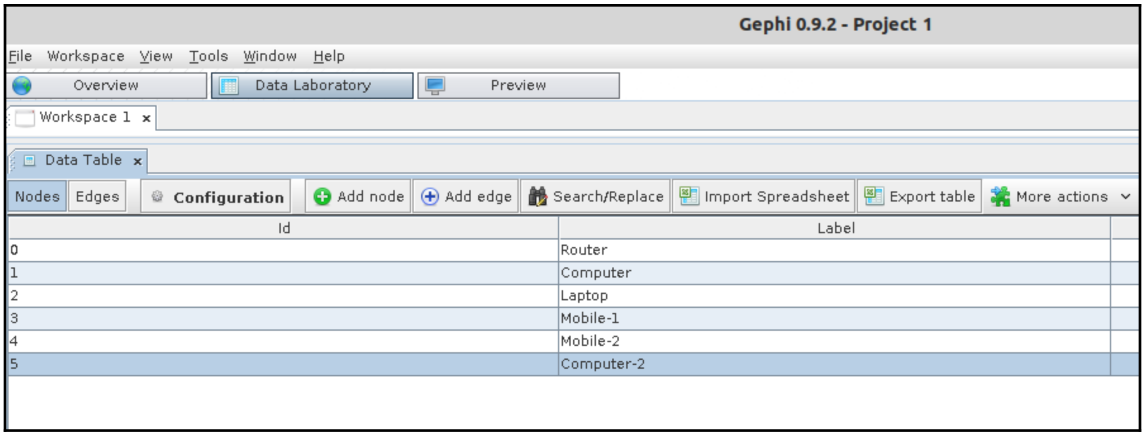
The Add Node button location in Data Laboratory

4. Once you click on **Add Node**, it will pop up a dialog box so that you can provide a name for your new node. In our case, we will name our first node as **Router**:



Adding a node in a network using Data Laboratory

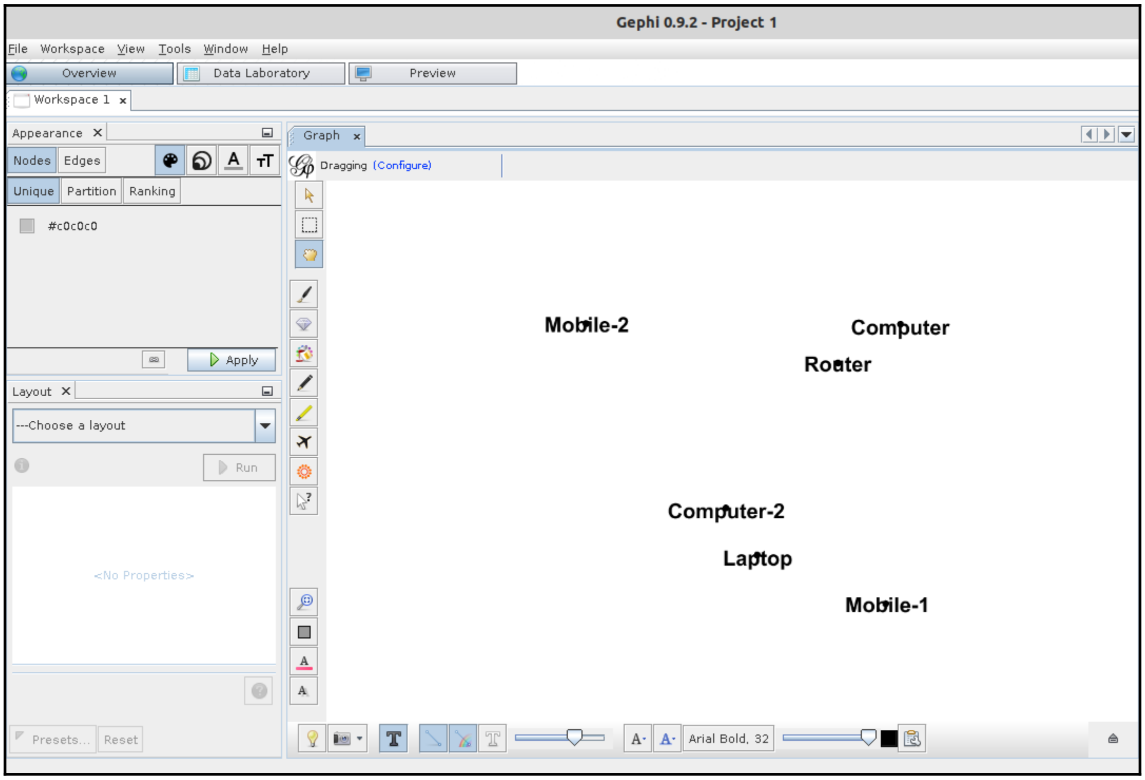
5. Similar to the last step, after adding the **Router** node, also add the following nodes, as shown in the following screenshot:
    1. **Computer**
    2. **Computer-2**
    3. **Mobile-1**
    4. **Mobile-2**
    5. **Laptop**



Id	Label
0	Router
1	Computer
2	Laptop
3	Mobile-1
4	Mobile-2
5	Computer-2

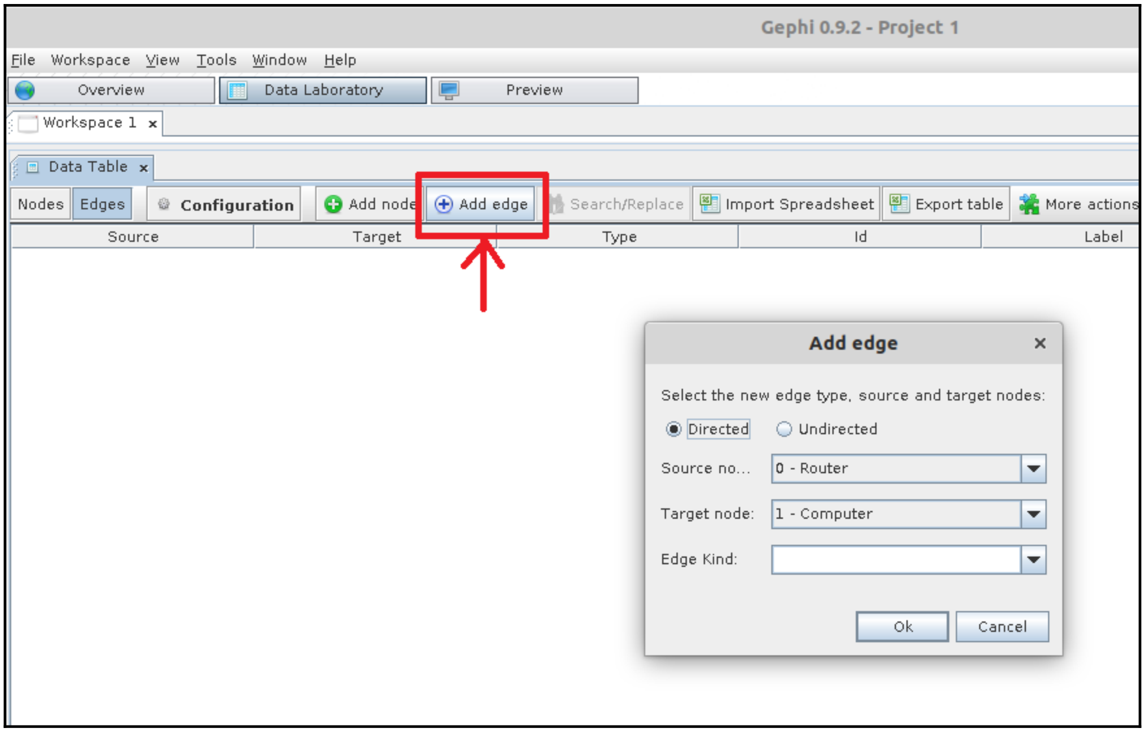
Complete node configuration in Data Laboratory

6. Once you finish adding the required nodes, click on the **Overview** button from the top left menu navigation, similar to what we did to switch to **Data Laboratory** mode. It will visualize the current network setup, as shown in the following screenshot:



Node overview on network diagram

7. It's now time to connect these nodes, so that we can establish how these nodes are connected to each other. Click on the **Data Laboratory** button again from the top left menu to switch the mode. Now, click on the **Add edge** button to start establishing the links, as shown in the following screenshot:



Select the new edge type, source and target nodes:

[✓ Directed] Undirected

Source no... 0 - Router

Target node: 1 - Computer

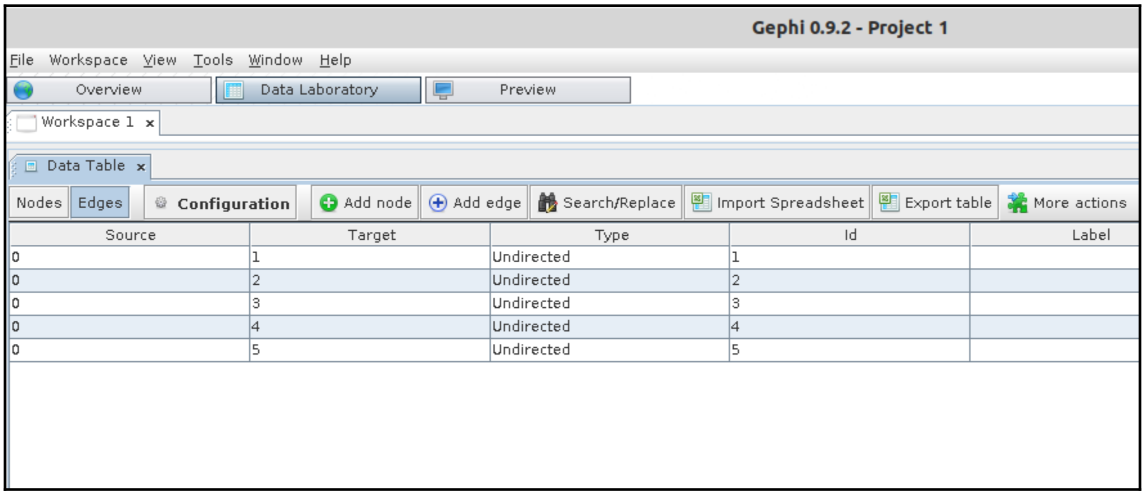
Edge Kind:

Adding an edge in a network

8. In our sample network, all the nodes are connected to Router. Select the edge type to be `Undirected` because it is not a unidirectional connection. Now, select the `Source` as `Router` and the `Target` as `Computer`. Then, click **Ok** to add edge information in our **Data Laboratory**. In a similar fashion, add the remaining edges, as shown in the following table:

Source	**Target**
Router	Computer
Router	Computer-2
Router	Mobile-1
Router	Mobile-2
Router	Laptop

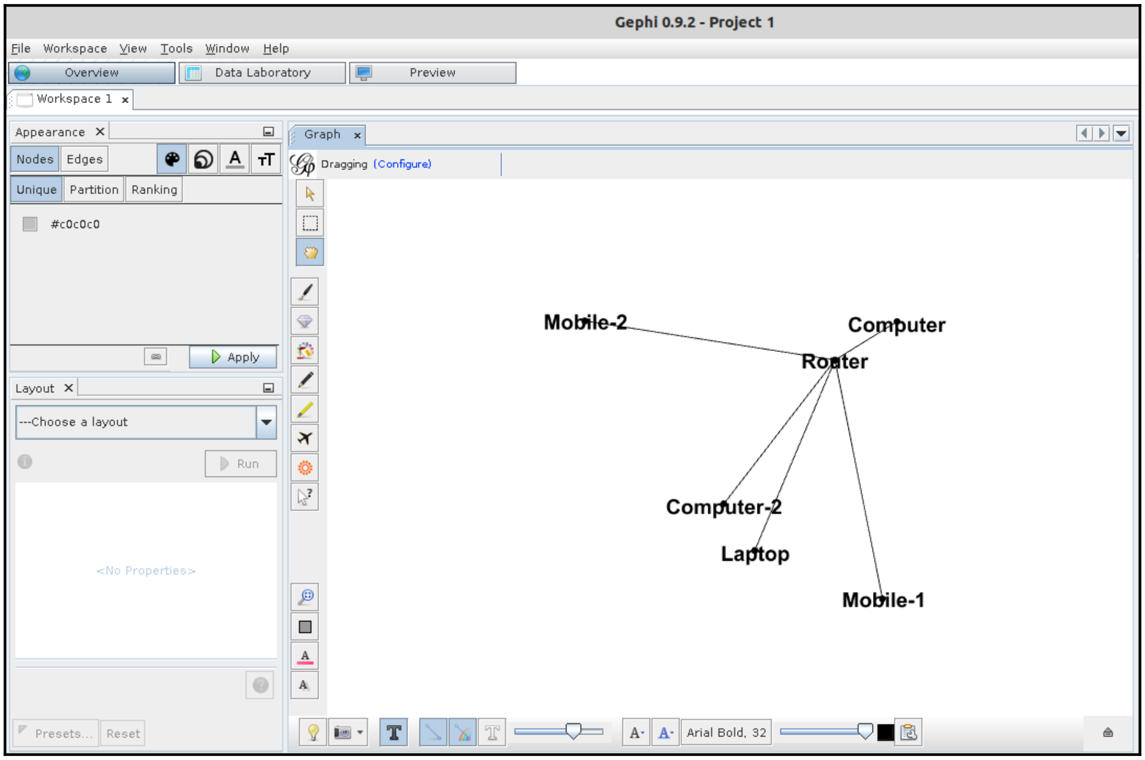
9. The following screenshot illustrates a similar configuration in the **Data Table**:



Source	Target	Type	Id	Label
0	1	Undirected	1	
0	2	Undirected	2	
0	3	Undirected	3	
0	4	Undirected	4	
0	5	Undirected	5	

Network edge complete configuration

10. Now, to visualize the network that we have established, click on the **Overview** button from the left top menu as we did earlier. The following screenshot displays the output of our network:



Full network visualization

Now, imagine how easy it is to visualize all the objects in this network and how they are connected to each other. This example is performed on a small network, but if you apply it on a large network setup, it will still be very easy to understand the network along with all network objects.

Summary

A well-designed network leads to a high performance infrastructure with better utilization of its resources. In this chapter, we have learned about the different types of network configuration, such as local area network, metropolitan area network, and wide area network. We have also learned about the different connectivity methods used to established these types of networks. We have also learned about the different bands used by wireless connections.

We then moved on to the more practical side of a network, designing and using a tool named Gephi to help visualize a network. We used a sample local area network scenario where we took two computers, two mobile phones, and a laptop that were connected to each other with the help of a Router. We went through a step by step guide to build the network diagram in Gephi, which will help to visualize the network for better analyses.

In the next chapter, we will go through different design considerations for when you are designing a big data cloud infrastructure.

8
Cloud Infrastructure

There are lots of companies that are now moving their businesses and IT infrastructure to cloud providers, either completely or in part. They are reaching this stage because they are convinced that it is the future in terms of infrastructural setup. A survey was conducted that concluded that by 2019 65% of businesses will have moved to cloud.

In this chapter, we will be covering the following topics:

- Companies moving to cloud
- Infrastructure driving factors
    - Data locality driving factors
    - New requirements driving factors
- Design consideration
    - Open source versus commercial
    - Commodity hardware versus purpose built
    - Cloud versus on-premises
    - Application architecture
    - Cost

Let's first start with some discussion on the driving factors due to which companies are moving to cloud infrastructures.

Companies moving to cloud

Companies are always looking to ease things up. With the evolution of cloud service providers has taken away the headache, from small companies at least, of bearing huge startup costs, in terms of building their IT infrastructure. Because most startup companies have low budgets, instead of investing their budgets into the IT infrastructure, they always want to spend money on their primary business in order to grow it.

Now, with so many cloud service providers and the varying degrees of services they provide opens up a whole new market for themselves as well as for the businesses. In the subsequent sections, let's discuss some of the factors that drive companies to make this decision to opt-in to a cloud-based IT infrastructure.

Driving factors

The driving factors that make companies move to a cloud-based infrastructure can be divided into three main sections, which are as follows:

- Infrastructure
- Locality of data
- Requirements

Let's start our discussion with the first section, *Infrastructure*.

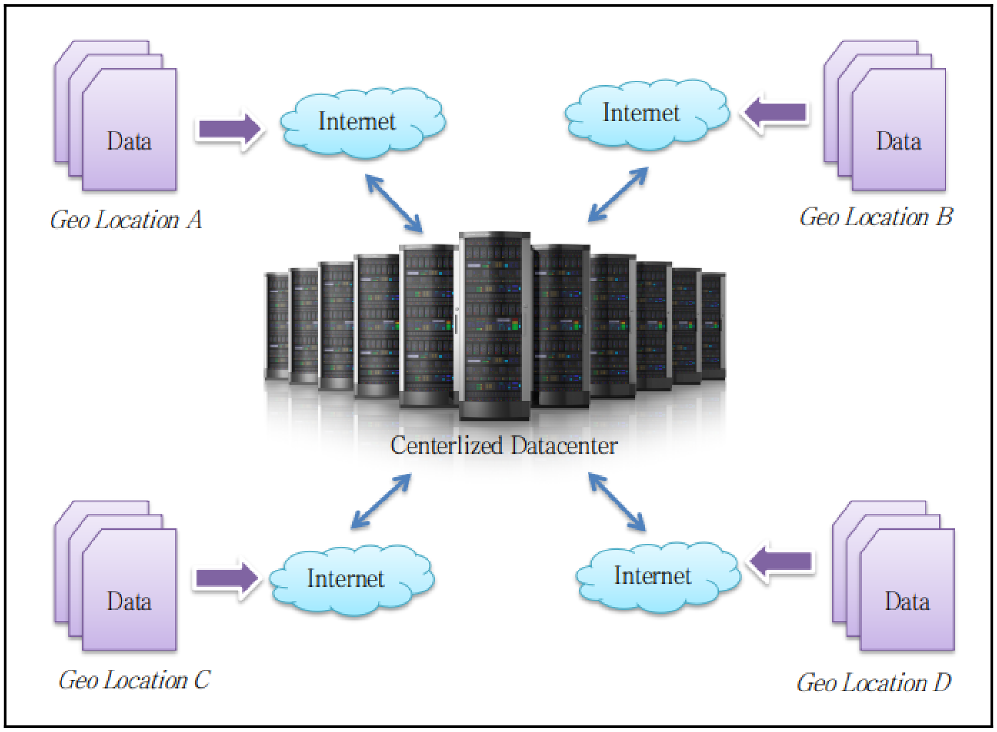
Infrastructure

Let's suppose a company has an on-premises data center that contains 10 servers. Now, due to any reason, you want to add a new server to your data center. Imagine the paperwork and approval process involved before you get the new server. Getting hardware requirements from the concerned stakeholders, then getting approval on that, then requesting different vendors to provide their quotations based on the provided requirements. Once you get the quotation, you have to get it approved by the higher management. Once you get the approval, you have to issue a purchase order to the vendor to provide you with the goods. Then, it will be passed on to the IT team to physically install it in your premises. Later on, they will do the configuration and setup as per their guidelines. These are some of the steps involved, which vary from company to company, and this process will take no less then five to six months to complete.

Whereas, on the other hand, once your business is on the cloud, all you need to do is to collect the requirement as to which type of server is required, get the approval from higher management about the related cost and that's it. If you stretch this process too much, it will still not take more then a month to complete. It will then be just a matter of a few clicks from your cloud infrastructure management console to add a new server into your network. And the best part is, if at any point of time you don't need that server anymore, just remove it from your cloud management console and you will not be charged for it any longer.

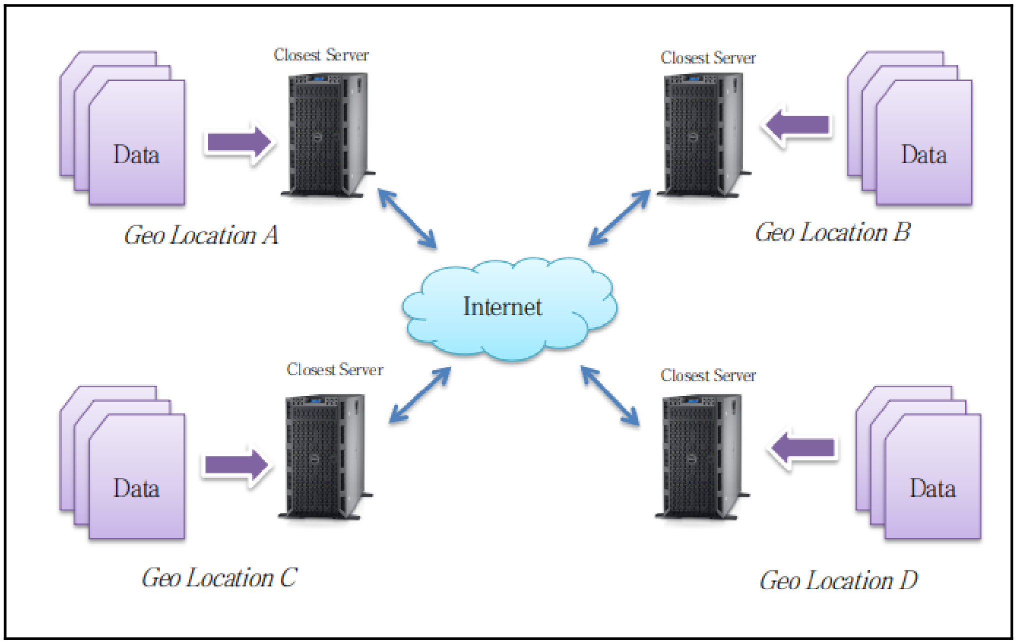
Locality of data

Nowadays, most businesses are spread across multiple countries in different geographical locations. Now, having an on-premises data center in one part of the world and customers and devices that are using that data center in another part of the world doesn't seem to be good practice. It will face data latency issues along with the overhead of the additional data transfer cost. It will result in an average to low-grade service with extra cost. The following diagram illustrates this type of architecture:



Single-premises-based data sharing architecture

On the other hand, cloud infrastructure providers have their data centers spread over different geographical locations on different continents. Having this type of setup leads to reduced data latency rates and increased performance. So, instead of keeping our servers in one place and letting all the data come to it, we will divide our servers into different locations. We will keep the servers near to the location from which the data is being generated. For example, if our main clients or devices are in China, then we place our servers close to China, most likely within Asia. And, because it is a cloud infrastructure, you don't need to worry about all the trouble that comes with placing your server close to your serving location. The following diagram illustrates the cloud-based data locality architecture:



Cloud-based data locality architecture

Requirements

We are living in a technologically advanced era, where researches and inventions are being done on a very rapid scale. Every now and then, we are being presented with new software, hardware, IoT, and many more things. Similarly, to adopt and cater to the requirements of these new inventions, your data centers should be needing constant upgrades, whether it's a hardware upgrade or a software upgrade. Plus, in a traditional system, we store only data that is important to us, but now that we understand the power of data, we should have the capacity to ingest all types of data, whether it is useful for us or not, and it can be of any type, structure, semi-structure, or unstructured.

These types of requirement changes lead to constant changes in your on-premises infrastructure, and will also financially affect you. Whereas, with a cloud infrastructure, you are not purchasing any hardware, you are only renting it, and only pay for the amount of time you used it. So today, if your requirement is different from the requirements of tomorrow, it only requires some configuration changes, based on how your usage fees may increase or decrease, and that's it. You will have a full flash new infrastructure ready to go.

Now that we have discussed some of the driving factors that are moving companies from having their on-premises infrastructure to having a cloud infrastructure, let's move our discussion to different design considerations that you should keep in our mind while designing and setting up your application to be used on a cloud infrastructure.

Design considerations

Now that we have discussed some factors due to which companies are moving their infrastructure onto cloud, let's now discuss some key factors that one should look at while designing an application to be used in this type of infrastructure. Applications are now more advanced and utilize different types of data instead of the traditional application, which uses structured data either in the form of a relational database or local storage. The following are some of the key considerations that one should cater to before designing a cloud-based infrastructure and the applications:

- Open source versus commercial
- Commodity hardware versus purpose built
- Cloud versus on-premises
- Application architecture
- Cost

Let's start our discussion with the first topic, scalability.

Open source versus commercial

Now, while designing an infrastructure to run your applications on, you may need to decide which type of setup you want to have; either you want to use open source tools or instead you will use commercial tools.

The open source community is more like do it yourself. So if you are opt-in for open source tools and technology, keep in mind that you have to play around with it and do the stuff by yourself. The open source community is growing very rapidly and more and more people are getting in to play around with the stuff, ready to help others if any sort of support or help is needed. So, your main point of contact to resolve any issue or to get technical support is the community. The main advantage is that you have the full code of the tool or technology you are using so you know what's in there and how to manipulate it to meet your desired requirements. Obviously, it requires more hard work from your end in terms of time and effort, but in the end you will get a full flash open source tool without any recurring costs.

On the other hand, if you opt-in for commercial build tools or technology, you will get a fully tested, quality assured, and compatible system that is ready to be plugged in for your use. In general, commercial tools have their custom functions available along with the tool, which is an added advantage. It comes with all the support you need to resolve your issues, as most vendors have dedicated teams to cater any client issue available 24 /7. The downside is that it comes with a price tag. The bigger the vendor, the higher the price tag; similarly, the price tag also depends on the complexity of the tool required. More functions means more cost.

Commodity hardware versus purpose build

Now, when it comes to the hardware, there are two options. The first one is to call up vendors and give them your specifications or requirements. They will then come up with some premium quotes. You will then select one of them to do the hard work for you and get you infrastructure ready. Obviously, it will cost some stash of money. The advantages you will get from it are that you will get a engineered build infrastructure, closely coupled with all the components with in it, from the operating system to all the micro components, including hardware and software. You will also get full time vendor support, so if you face any issues they will be there to help you out and resolve it.

With in the vendor category, you also have the option of going for the cloud infrastructure. You will have all the advantages of the vendor custom engineered setup, but instead of purchasing the full infrastructure, you will be renting out the setup based on your utilization. If you need only 5 servers from the full grid, you will have to pay only for that, but in a timely fashion, as you will be renting it, not purchasing it. There are many different types of configuration available, from renting a full flash server to just renting the computing power or just renting storage capacity.

On the other hand, if you opt-in for commodity hardware, then it's up to you what type of hardware you will use to build your computing grid. You can mix things up, such as by having some hardware with more computing power and some with a larger storage capacity. It all depend on what your requirements are and how you pursue them. You can start small and gradually build on it one piece at a time. There are many grid management console tools and applications available in the open source market to help you out. The drawback is that you will have to do it by yourself; you will not get any specialized people to do the job for you. The main advantage is that it will have very minimal effect on your bank account as the components and hardware you will be using are mostly cheap.

Cloud versus on-premises

Let's now talk about building the infrastructure on-premises versus going for a cloud-based infrastructure solution. If you set up the infrastructure on-premises, the main advantage you will get is all of your data utilized within will stay local, within the boundaries of your premises. You will not be having debates about how secure your data is and who else will have access to it. You will have more control over it and all the other components involved in the infrastructure. Once you purchase the hardware, you will have the option to either run it on full throttle using maximum computation power or leave it idle. It's yours to use with no money involved. The downside is that you will have to bear the cost of a team that will handle all the infrastructure setup and maintain it.

The following diagram illustrates the on-premises setup:

Illustration of on-premises setup

On the other side, if you choose to go with the cloud solution, then you will bypass all the initial heavy costs for the infrastructure to get you started. Instead, you will be paying only for the resources you will use of the cloud provider. In simpler words, you will rent the infrastructure from a cloud provider based on your requirements and you will pay for it. There are couple of advantages to this approach. First of all, you will not be barged in with heavy duty bills, seeing instead small amounts depending on the type of resource you rent. The other advantage is that you will not have to bare any maintenance costs for the cloud infrastructure either, as you will not be the team responsible for the task. Instead, it is the cloud provider's responsibility to keep the things on track. One of the important advantages is that you can have your infrastructure placed in different geographical locations so that the computation and the storage will be closer to the place where it has to be utilized.

The disadvantage of this approach is that you will have less control over the privacy of your data. Because the data is out there, you may not know who has access to it. Of course, you can take precautions to prevent the data leakage, such as encrypting you data, applying more security on the server you rent, and so on, but still, you will always have some doubts that your data is out in the cloud. The other disadvantage is that if your data needs heavy duty computation power, that the bills that your cloud provider will send you will definitely surprise you. You will need to keep a close eye on the amount of resources and the computation power you used in this type of approach. The following diagram illustrates the cloud-based infrastructure setup:

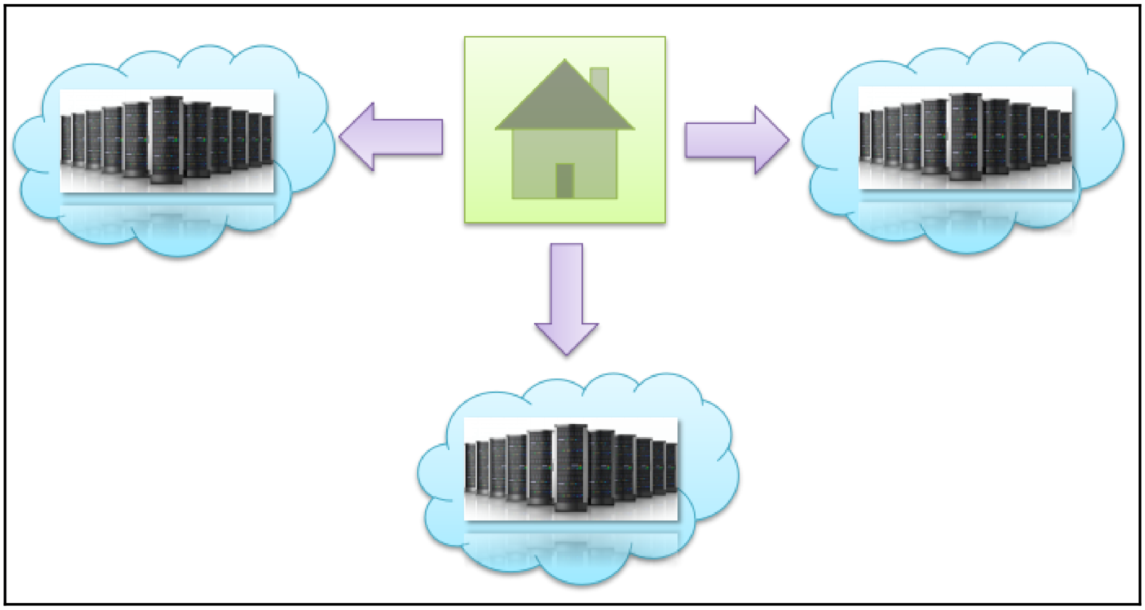


Illustration of cloud-based setup

To cover up the disadvantages of both on-premises and cloud-based setup, some of the companies are now opt-in for the hybrid solutions. In this type of setup, they keep some of the infrastructure within the premises and some of it out in the cloud. On-premises infrastructures mostly consist of storage for the sensitive data, along with high computational power so that most of the computation-expensive operations can be done on-premises. The cloud infrastructure can be use to handle IoTs, marketing, end-user applications, and so on – the sort of data that is less computation expensive – and to get the infrastructure close to the location where it is being utilized. The following diagram illustrates the setup of the hybrid approach:

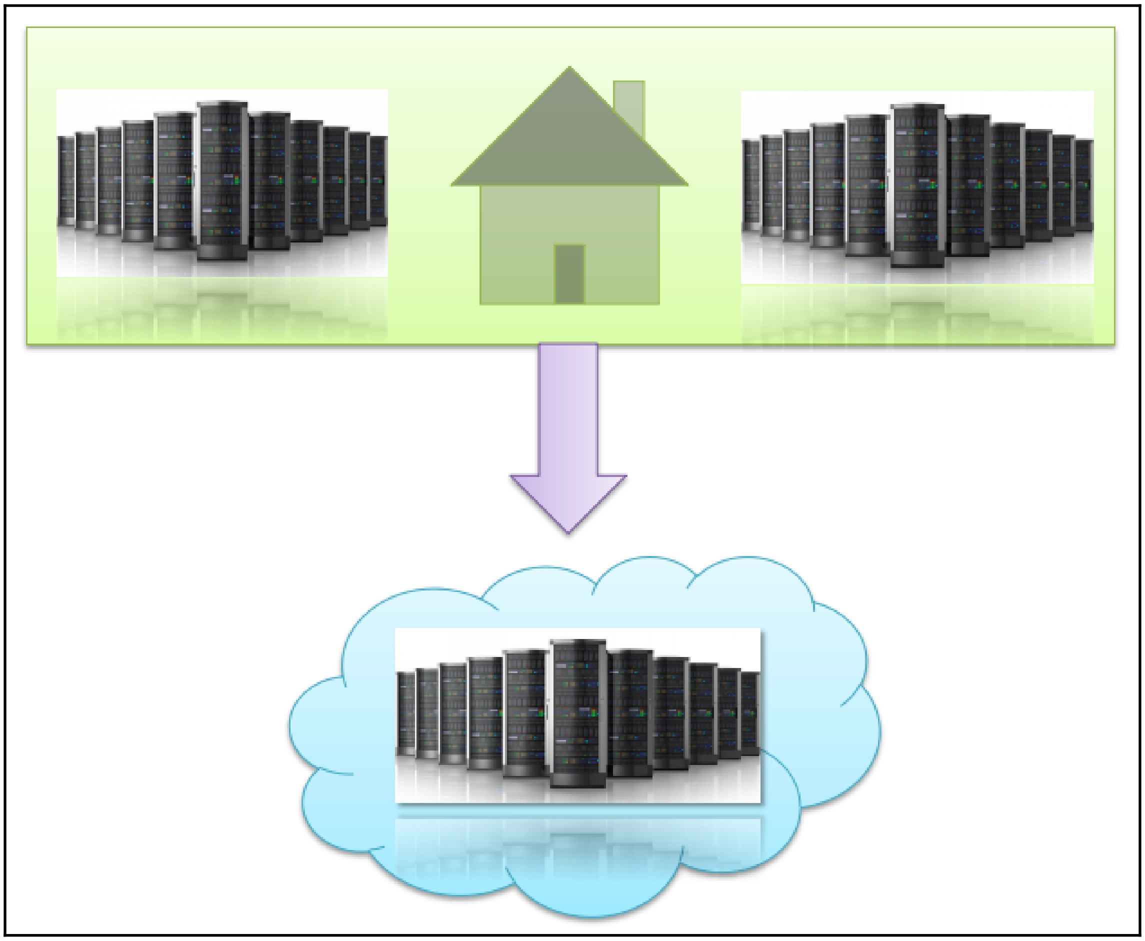


Illustration of hybrid infrastructure setup

Scale up and down

In case of on-premises infrastructure setup, when you have a new requirement, it is most likely that you have to add a new machine to your grid, because in most cases you will have built your infrastructure based on the requirements that you currently had. So, every time you get a new requirement, you will buy the required hardware and add it to your grid. It is very likely that you will have commercial support that will slap you with some premium quotes for it, which in the end you have to bear in any case. It may go through some hectic approval processes, but, still, it has to be done. This is in the case where you have to scale up. In case of scaling down, you don't have any alternatives; you end up with a resource that you are not utilizing. It may be used in future, but, for now it's a part of your grid. The following diagram presents the illustration of scaling up your infrastructure:

Illustration of scaling-up your infrastructure

On the other hand, if you have a cloud-based infrastructure, most cloud service providers give an option to auto scale your jobs so that if today your computation job has less data, which may utilize the computation power of five servers, then you will only be charged for that. Similarly, tomorrow, if you have more data and you need the computational power of ten servers, then the provider will automatically scale the job to ten servers and you will be charged only for that.

Similarly, if you have ten servers from your cloud provider and you have a new requirement to deploy a new application on a separate server, this can be handled very easily from the cloud management console; in a matter of a few clicks, your new server is up and running. Now, if you decommission any applications and you don't need the server that it was running on anymore, simply remove it from your infrastructure, and you will not be charged for it anymore. The following diagram presents an illustration of scaling-down your infrastructure:

Illustration of scaling-down your infrastructure

Application architecture

One of the important factors to consider is the application itself that you will be running on the grid. Whether the grid is on-premises or on the cloud, if the application is not built to run on your distributed environment, then you will not be able to utilize the full potential of your grid. For example, you have 10 servers in your data center, five servers with high computational power, and five servers with a high storage capacity. Now, you bring an application to run in this environment, but the application is built to run on a single machine. In this case, you will end up with either the high computation power of a single machine with less storage capacity, or the high storage capacity but low computation power of a different single machine. In both cases, you will be at a loss, as only one of your machines is being used, and the other nine are sitting idle, waiting to be utilized.

Cost decision

The essence of everything comes down the cost. If you have the budget, you can buy a Lamborghini, but if you don't have the money then even a cheap item is hard to get.

Normally, startup companies, or small companies, have tight budgets. Instead of spending the money on secondary things, they always look to invest in the primary thing. In traditional business models, setting up an IT infrastructure is a must, and there are no escape routes available. In their worst condition, they end up sharing the infrastructure resources among multiple companies.

Now, with the evolution of cloud service providers, they now have the option to at least rent out the required resources according to individual requirements in the initial stage, and, later on, if they want to have an on-premises infrastructure, they can also do it.

There are many cloud service providers that offer a range of services, from dedicated servers, to shared servers, storage, web hosting, and so on, with a price tag. The following is a list of the top-10 cloud service providers:

- Amazon Web Services: `https://aws.amazon.com/`
- Microsoft Azure: `https://azure.microsoft.com/`
- Google Cloud Platform: `https://cloud.google.com/`
- Adobe: `http://www.adobe.com/`
- VMware Cloud: `https://cloud.vmware.com/`
- IBM Cloud: `https://www.ibm.com/cloud/`
- Rackspace: `https://www.rackspace.com/`
- Red Hat: `https://www.redhat.com/en/topics/cloud`
- Oracle Cloud: `https://www.oracle.com/cloud/`
- Verizon Cloud: `http://www.verizonenterprise.com/welcome-to-verizon-cloud/`

Some of them offer free trial versions for you to explore the functionality they have to offer. For example, Amazon, Google, and Microsoft are offering up to 12 months of their free trial versions, and on the first month up to $200 credit can be used on some of the premium services they are offering. Go ahead, sign up for their trial accounts and start exploring the cloud.

Summary

In this chapter, we discussed the key driving factors that are making businesses decide to move to cloud-based infrastructures. Some of the key factors that we discussed in detail include infrastructure, locality of data, and the fulfillment of new requirements. Especially for small to mid-range companies, it is very hard to ignore these factors while setting up their IT infrastructure.

We then moved on to discuss the design considerations that you should consider while making decisions related to setting up IT infrastructure. One of the key considerations that we discussed in this chapter is the comparison of open source to the commercially available tools and technologies. We then discussed the utilization of commodity hardware against purpose-built hardware. After that, we moved on to discuss the advantages and disadvantages of on-premises and cloud-based infrastructure. We also discussed some of the primary considerations related to the application architecture that we will be running in our distributed environment. In the end, we concluded that cost is the essence of all the decision making.

In the next chapter, we will be going through some network monitoring and security tools.

9 Security and Monitoring

Network security and monitoring is always a big concern for companies. The size of the network was not that big in older day, therefore it was easy to manage. With the increased use of computer-related devices, the size of the network is exponentially increasing. Take an example of a data center, where each data center contains thousands of network devices serving their customers. When you have a big network that contains this many network devices, it is not possible for you to check whether the hardware and software components of each and every device are working properly. For this purpose, there are some standard network protocols and different monitoring applications that are open source, free, and commercially available in the market.

As far as security is concerned, until a couple of years ago, most companies restricted their network, which only allowed connected network devices to communicate with each other, without the interference of any direct or indirect connection from the outside world. All the servers and clients were part of the on-premises network setup. Even then, the companies were not satisfied with the security applied on them. Now, with the internet as a mandatory requirement, and most of the servers and information on the cloud, it is necessary for companies to expose their network to the outside world. This is something that is of great concern to many companies, and so they want the best security available. They always look to install different firewalls and many different types of network security and monitoring tools to make their network secure.

In this chapter, we will discuss the following topics, as they are related to network protocols, monitoring, and security:

- Simple Network Management Protocol
- Netflow
- Nagios
- Security Onion
- Wireshark

Simple Network Management Protocol

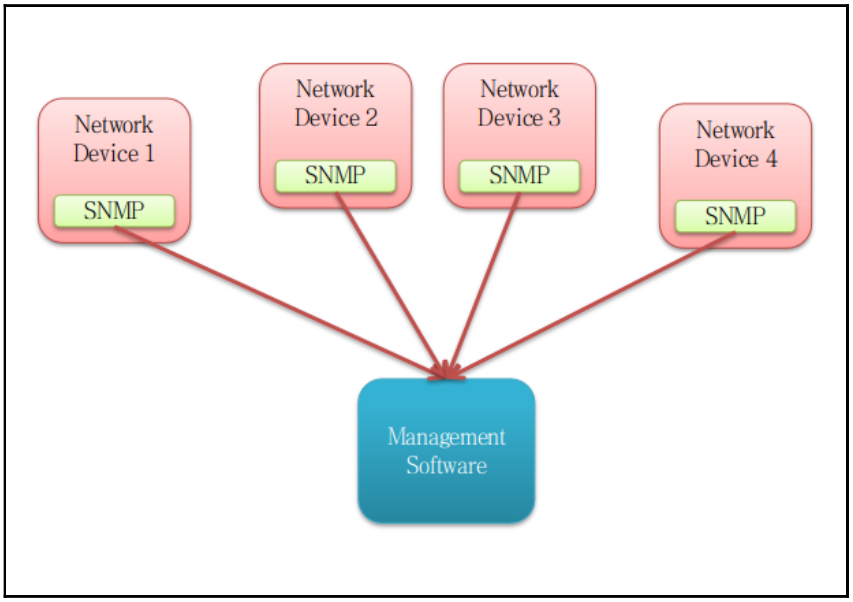
Simple Network Management Protocol is commonly referred to as **SNMP**. This protocol allows you to proactively view and analyze what's going on in your network devices, so that you can fix issues before they become a major problem. If you are in an environment where you can perform proactive maintenance, SNMP is a must-have protocol for you. Using this protocol, you can extract lots of information about the device.

SNMP is normally a component used as a part of an open source to enterprise-level application that helps you keep your network in order. Applications such as Spiceworks, LAN View, and Open Manager use SNMP to know about the devices in the network and gather related information from them.

In the standard monitoring solution, we use the default commands, such as ping, to check whether it is online, you can also use some other commands to check the IP address assigned to that device, and so on. But just by using those commands, you cannot know what's going on inside that device. SNMP comes to the rescue here and allows you to get the inside information about the device. For example, how much memory it is using, how much of the hard disk space is left, and what is the current CPU utilization.

Other than this, it can also let you know about the state of the hardware, such as whether the CPU fan is working, or whether all the assembled RAM are functioning properly. SNMP can send all this information back to a network-management software to highlight the current state.

The following diagram illustrates how SNMP works as a whole in a network:



How Simple Network Management Protocol works

Benefits of SNMP

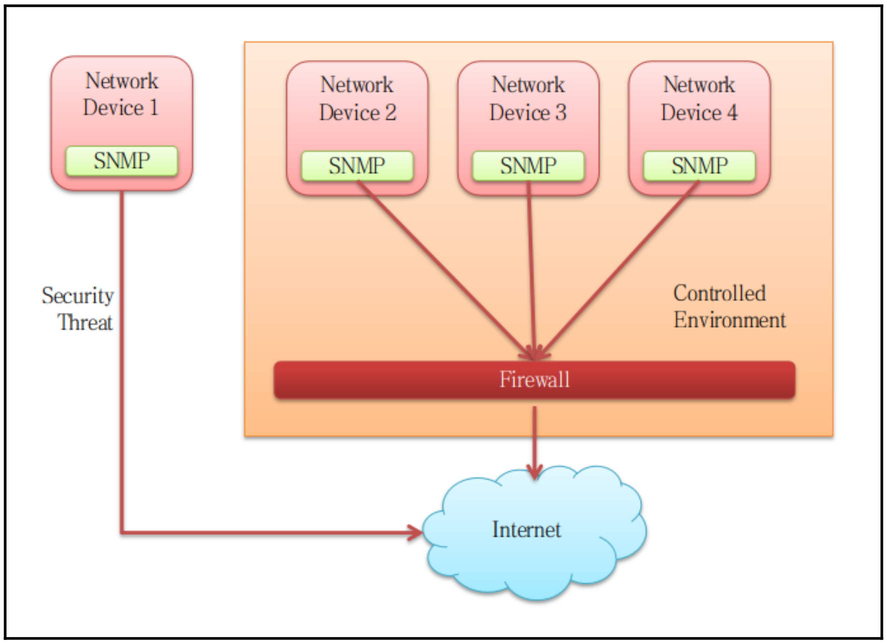
Let's now try to understand the benefits of SNMP. Assume you are an administrator in a small organization where you have 10 to 20 network devices to manage. It will not be too much trouble for you to monitor these devices and keep them up and running in a fully-functional state. Now imagine you have a data center of two hundred network devices. How will you monitor all of them? It is not possible for you to open up each of the network devices and check whether all of the components are working properly.

Now imagine in one of the network devices, the case fan just stops working. Eventually, the heat in that device will build up and it can destroy the entire device. It is a loss in itself but in some cases, it may catch fire, resulting in a much bigger loss of both property and, most importantly, data and the operations.

SNMP comes to the rescue in these types of situations. There is always a time window where you can take necessary actions before an issue becomes a big problem. In the event of any of such cases, such as hardware component failure or the utilization of any of the components exceeds the set threshold value, SNMP will immediately notify Management Software to take corrective action. This prevents any unforeseen issues or a bigger loss.

Security

SNMP can provide great deal of information about the network device. Back in the Windows XP era, many of the networks got hacked because of SNMP. It normally comes as a backend service, along with the operating system, and in Windows it is enabled by default. So if it is not properly handled in your network, it will open the back door for hackers to attack. The following figure illustrates this scenario:



Possible security threat due to SNMP

Nowadays, the first thing a network administrator does is to disable the SNMP service and enable it only when the required configuration and setups are in place, such as firewall settings.

Agents and Traps

When you work with SNMP and its related software, you will come across two common terms: **Agents** and **Traps**.

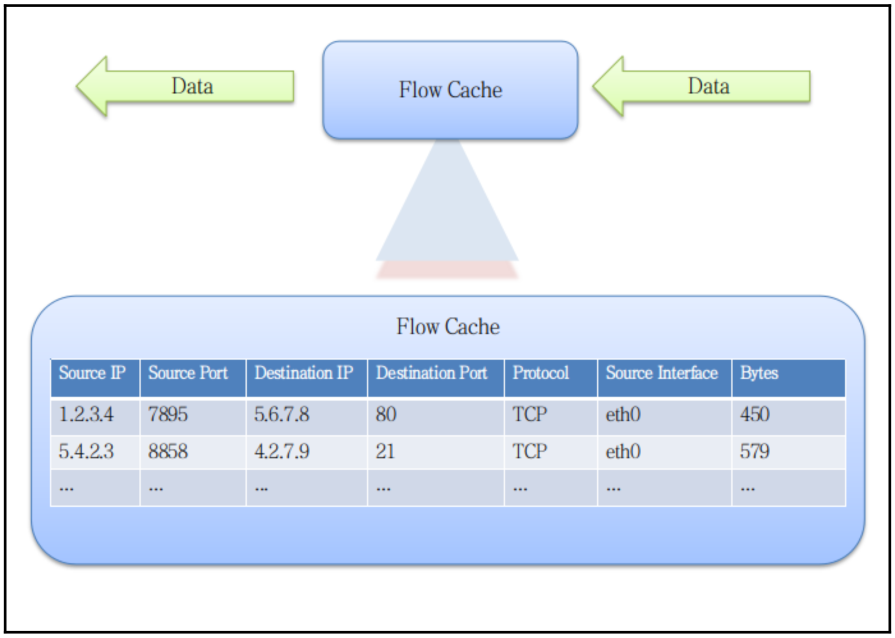
An Agent is a piece of software that runs on the network device that we want to be monitored. The purpose of this agent is to notify the central-management software in case any of the configured events happen, such as the hard disk utilization reaches 90% or one of the memory slots stops working.

The items that we configure, which are picked up by SNMP agent to monitor are called a Trap. For example, getting a notification upon the hard disk utilization is a trap, the failure of a memory slot is a trap, and so on.

Now that we have understood the basics of SNMP, let's move on to our next topic, Netflow.

Netflow

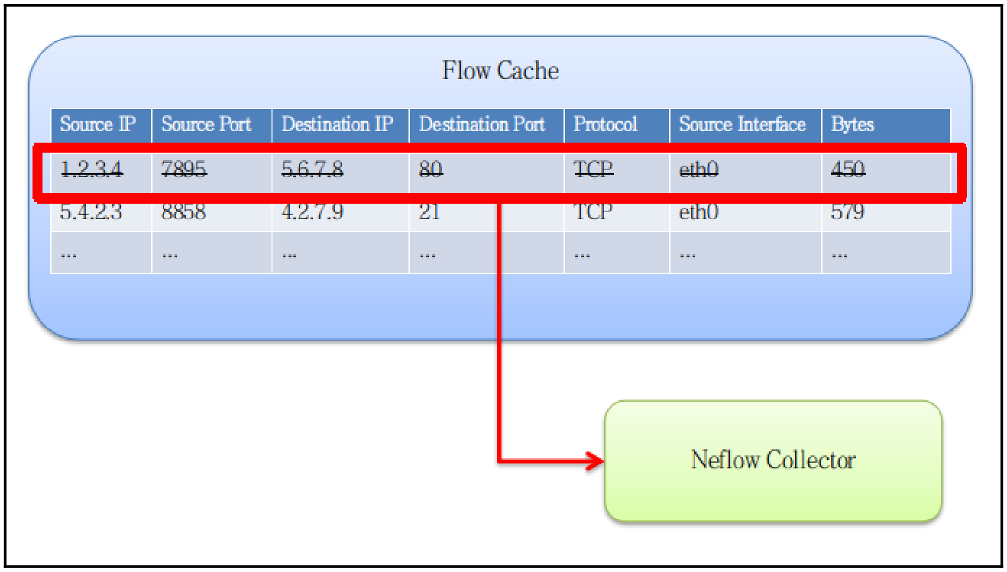
Netflow is a network protocol developed by Cisco. It is responsible for collecting data related to the network traffic that flows through it. It collects information such as Destination and Source IP addresses and ports, protocols used in communication, total data transferred, and the network interface user for this purpose. The following figure illustrates the architecture of Netflow:



Source IP	Source Port	Destination IP	Destination Port	Protocol	Source Interface	Bytes
1.2.3.4	7895	5.6.7.8	80	TCP	eth0	450
5.4.2.3	8858	4.2.7.9	21	TCP	eth0	579
...	...	...	...	...	...	...

Netflow architecture

The flow cache can contain records from a few thousands up to millions. It depends on the type of device that is being used for the Flow cache. After the record in the flow cache expires, it will then transfer that record to the Netflow collector for further analysis. The following figure illustrates this concept:



Source IP	Source Port	Destination IP	Destination Port	Protocol	Source Interface	Bytes
~~1.2.3.4~~	~~7895~~	~~5.6.7.8~~	~~80~~	~~TCP~~	~~eth0~~	~~450~~
5.4.2.3	8858	4.2.7.9	21	TCP	eth0	579
...	...	...	...	...	...	...

Netflow collector opertion

Netflow can be embedded in the device hardware or can be presented as a software package. It can be used in real time. It is not used just for network-monitoring purposes; it can also be used to do the following:

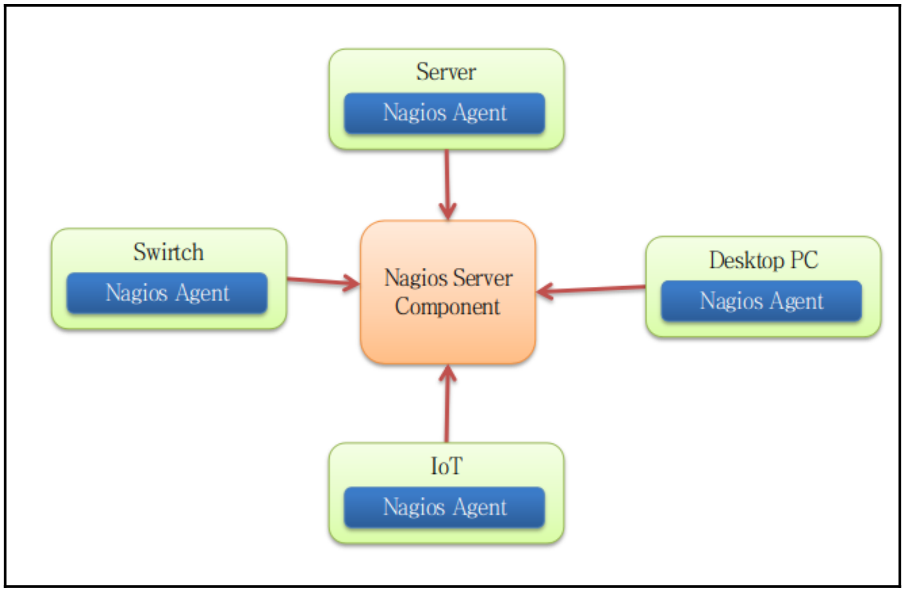
- Monitor a specific application's activity
- Monitor user activity
- Network design and planning
- User-based billing and reporting
- Application profiling
- Security analysis
- Bandwidth monitoring
- Cyber threat forensics

IPFIX is standard for any flow protocol. Other vendors have developed their own version of Netflow; HP developed s-Flow, and Juniper Networks developed J-Flow.

Nagios

Nagios is a free and open source network and IT infrastructure-monitoring application. It provides different types of monitoring options, including servers, computers, switches, and applications. It was initially named NetSaint as was first released in 2002.

It runs under the category of a server client application. A client application is required to be installed on the device you want to monitor, which will be configured to provide feedback to the server running its server component. The following figure illustrates the architecture and flow of Nagios:



Nagios flow and architecture

Let's now explore some of the key benefits of using Nagios.

Key benefits

Nagios provides the following key benefits:

- It monitors network services such as SMTP, POP, SMNP, and HTTP
- It can monitor different operating system including Windows and Linux
- Hardware monitoring to identify any sort of failure or issues

- It provides options to remotely review and manage through the Remote Management Console over SSH and SSL connections
- It also provides options for developers to create their own plugins depending on the custom requirements
- Multiple notification options, such as an onscreen alert, sending an email, or SMS if an issue happens
- It has the capability to define event handling for proactively resolving problems
- It can be integrated with the backend database for more custom details and analysis

These are some of the key benefits of Nagios. The plugin-based design makes it more adoptable so that if any of the custom requirements are not already covered by Nagios, you have the option to create a plugin for that functionality and add it to Nagios. The rest will be handled by itself.

Up to this stage, we were discussing how to monitor a network. For example, all the hosts in the network are up and running, and whether all hardware and software of the individual host is also working properly. We will now discuss the application that will allow you to understand and analyze the data packets flowing in your network.

Security Onion

Security Onion is a Linux-based distribution that is specifically designed to help you monitor your network and analyze any malicious activity. It helps to identify any intrusion, logs all the activities in your network, and provides valuable feedback.

The basic architecture of Security Onion is based on the server-client model. The agents are deployed on the client machines, they are also called **Sensors**. The job of the sensor is to sniff through the activities on the network on configured hosts and report it back to the Security Onion server. It provides an interface for an analyst to connect to its server from a client machine and execute queries and jobs for forensic purposes.

A Linux distribution means that it is an operating system specifically configured with different tools to focus on a desired task.

Deployment scenarios

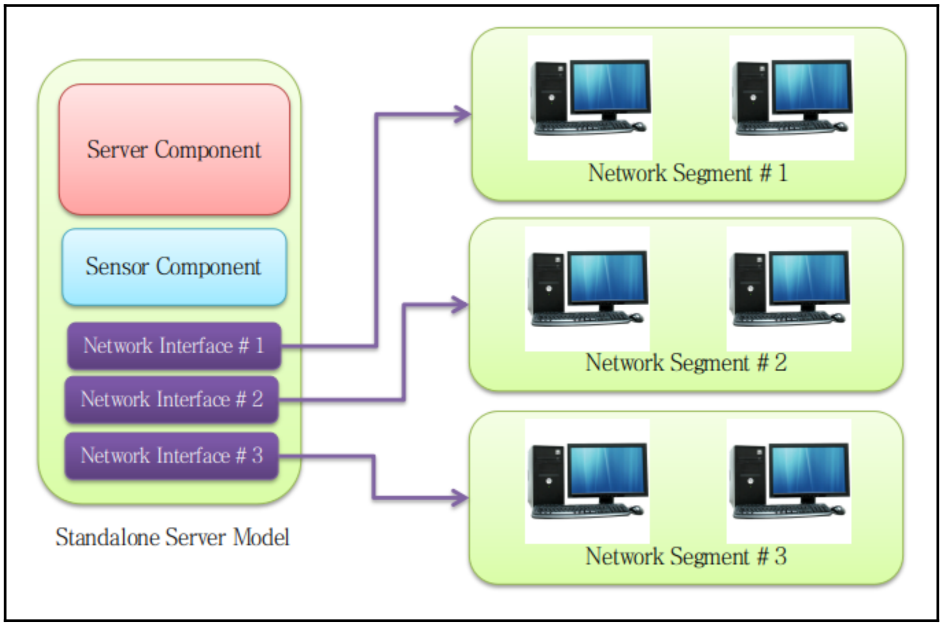
The Security onion model-based design allows you to configure it in three different ways:

- Standalone model
- Server-Sensor model
- Hybrid model

We will now go through all of these models and discuss their architectures and the differences between them. Let's now start with the first model, Standalone.

The Standalone model

In this type of setup, a single machine will be set up to run both the server and the sensor component, as well as related processes. The server component will be responsible for collecting data from the sensor and the sensor component will be responsible for sniffing the network using the configured network interfaces on that machine. You can set it up to work on different network segments by having multiple network interfaces on that machine. The following figure illustrates this mode:

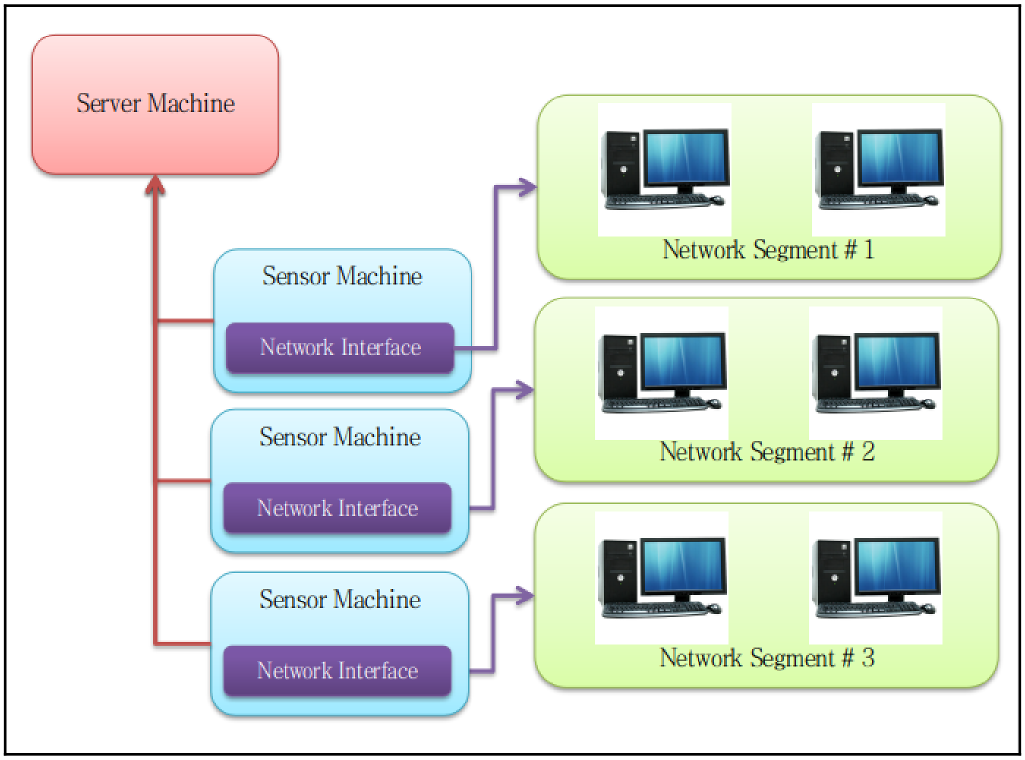


Security Onion Standalone model

This model is considered to be a more efficient and convenient model if you have access to the complete network from a single location. It makes the setup and deployment easy.

The Server-Sensor model

In this type of model, you will deploy the server component on a single machine, which will communicate the sensor components that may be deployed to one or multiple machines. Each Sensor component will be responsible for sniffing through the network on a set of preconfigured hosts or on a specific network segment, which then reports all the activities back to the Server component upon request. The following figure illustrates this model:



Security Onion Server-Sensor model

Once an analyst connects to the Security Onion server and queries any data, the server component will then request that data from the related sensor. This will result in reduced network traffic, as all the data will be kept with the sensor and will be provided on demand. All the traffic between the server component and the sensor is encrypted.

Hybrid model

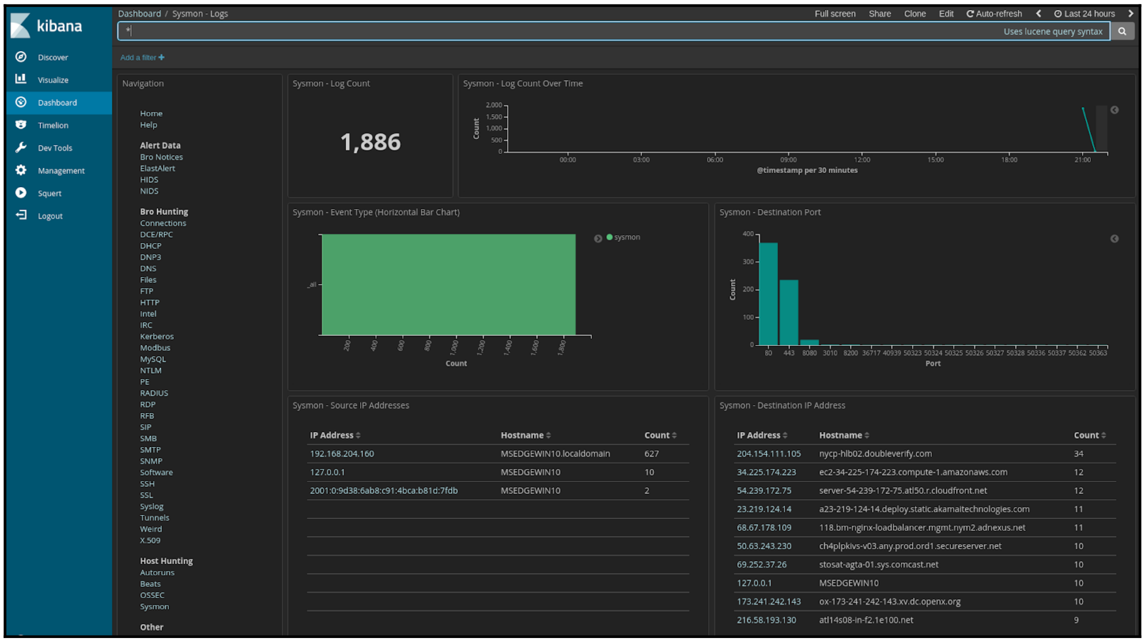
As the name states, it a mixture of both flavors. In this type of model, you will have a machine where the server component and the sensor reside, which work together, similar to the standalone model. Along with that, it will also have one or more sensors deployed on different network segments to report back to the standalone server component.

Preconfigured tools

It comes with many powerful tools that are used to handle all sorts of data and perform analysis on them. Some of them are as follows:

- Elasticsearch
- Logstash
- Kibana
- Squert
- Sguil
- Network Miner

Some of these tools are covered in this book with respect to Big Data, some of them we have already discussed and some of them will come in later chapters. They are being used in this Linux distribution as they provide a more efficient way to handle data, whether it is structured or unstructured. Not only that, tools such as Kibana will give you graphical insight in the form a dashboard. The following figure will give you taste of the type of reports and the dashboard expected from it:



Sysmon - Source IP Addresses

IP Address	Hostname	Count
192.168.204.160	MSEDGEWIN10.localdomain	627
127.0.0.1	MSEDGEWIN10	10
2001:0:9d38:6ab8:c91:4bca:b81d:7fdb	MSEDGEWIN10	2

Sysmon - Destination IP Address

IP Address	Hostname	Count
204.154.111.105	nycp-hlb02.doubleverify.com	34
34.225.174.223	ec2-34-225-174-223.compute-1.amazonaws.com	12
54.239.172.75	server-54-239-172-75.atl50.r.cloudfront.net	12
23.219.124.14	a23-219-124-14.deploy.static.akamaitechnologies.com	11
68.67.178.109	118.bm-nginx-loadbalancer.mgmt.nym2.adnexus.net	11
50.63.243.230	ch4plpkivs-v03.any.prod.ord1.secureserver.net	10
69.252.37.26	stosat-agta-01.sys.comcast.net	10
127.0.0.1	MSEDGEWIN10	10
173.241.242.143	ox-173-241-242-143.xv.dc.openx.org	10
216.58.193.130	atl14s08-in-f2.1e100.net	9

Kibana dashboard

Let's now move on to a tool that will help you to not only capture the data packet in your network but also look into the details of the packet and its contents. The tool we will discuss now is called **Wireshark**.

Wireshark

Wireshark is an open source application used to analyze the data packet in your network. It can be applied in multiple ways, for example, network troubleshooting, security analysis, and network forensics. It was originally named Ethereal but was changed to Wireshark in 2006. It is a cross-platform software that can run on Microsoft Windows, Mac, different Linux distributions, and Unix-based systems. It is a GUI-based application, but it is also available to be used in terminal mode, by the name of *T-Shark*.

Let's now discuss some of the key features of Wireshark that make it popular with network administrators when it come network data packet analysis.

Key features

The following are some of the key features of Wireshark:

- Wireshark is an advanced application that understands different network protocols. It can identify the structure of the data packet and extract the available information from it.
- It can parse different fields out of the data packet, as per the standards of the networking protocol.
- It can analyze data packets from a live stream or from an already-captured data packet file.
- It supports data capture from different network interfaces, such as Ethernet and Wireless.
- It supports both GUI- and command-line-based interfaces to analyze the data packets.
- It provides many filtration mechanisms to filter out the desired data packet from the live or already-collected data packets.
- It can automatically classify may different types of data packets, such as HTTP and VOIP.
- It provides options to color-code the desired information based on the defined rules.

It is free to download and will take your networking carrier to the next level.

Summary

In this chapter, we went through different network monitoring and security protocols and tools. We started by discussing the most common protocol used in the monitoring world, SNMP, Simple Network Monitoring Protocol. We discussed how it works, its key benefits, its security-related issues, and common terminologies used while working on this protocol. We then moved to another type of protocol used to monitor the data packets flow in your network. There are different vendor-based variations of this protocol; what we discussed here is developed by Cisco and called **Netflow**.

Then, we discussed the different monitoring applications that will help you not only store the network-monitoring logs and related data, but also represent then in a graphical manner. We started off by discussing a free, open source tool named Nagios. We discussed how it works and its key benefits. We then moved on to the fully configured Linux-based distribution that comes with a bunch of tools and utilities, giving you complete control over monitoring your network. This Linux distribution is called **Security Onion**. We discussed its architecture and different deployment scenarios, and the preconfigured tools that come along with this distribution, for example Kibana and Elasticsearch. Afterward, we discussed Wireshark; this is a very important tool used by many security professional to analyze the contents of the captured data packet. We discussed how it works and some of its key benefits.

In the next chapter, we will use practical examples to discuss the tools and technology used to design a frontend software architecture.

10
Frontend Architecture

The user interface of any web application plays a vital role in either making it a success or dragging it toward failure. If the user interface is friendly and responsive, users of that application will interact and play around with it for longer period and also revisit it many times. But if the interface is complex and lags on different occasions, it makes the user frustrated and more likely to move away from that application. It doesn't matter whether the application has very strong backend architecture and database support; these things are irrelevant to the end user. The end user needs a very good-looking, friendly, interactive, and responsive application at their end; with this, any application can be on the road to success.

In this chapter, we will cover the following libraries in detail, along with their key concepts, with respect to building a high-end frontend interactive architecture:

- React JS
- Redux

React JS library is used to build a responsive and component-based architecture. It comes with a Virtual DOM concept. The beauty of it is that it will update only the elements of the application that receive changes rather than refreshing the whole web page. Hold on to it; we will explore more over the course of the next section.

Redux library is used to maintain the state of the required elements in a global location of an application by introducing a single place of truth concept. We will learn more about Redux in its respective section. Let's go ahead and start with React JS.

React JS

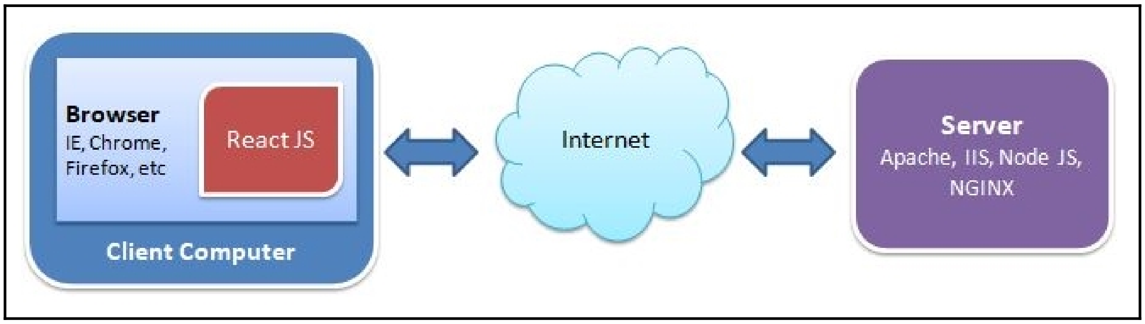
React JS is a JavaScript Library for building an interactive user interface with fast rendering. It is built by the Facebook company. If we look into React's history, we see that, a few years, Facebook came across a problem related to its interface; it was not very responsive and lagged while displaying multiple feeds. To resolve its user interface issues, Facebook came up with this library.

React JS Library is used only to fulfill frontend design requirements. It is just involved with the view part of MVC architecture and doesn't support any server, web service, or API calls. To do backend tasks, such as API or web service calls, database manipulation, server file handling, and so on, you need to use some other library or framework, such as JQuery. We will cover the backend architecture in our next chapter.

Now, to understand React fully, let's go ahead and first understand its design and the key concepts and terminologies used in our later discussions.

Key concepts

Similar to any framework and libraries, React also uses technologies and terminologies to understand its concepts and functioning. The following diagram illustrates the architecture of a web request and where React sits and renders the requests made to it:



Web Request Architecture and ReactJS's place in it

If we look into this architecture, a client computer makes a request to the internet, which is usually done using a browser. It can be any browser: Internet Explorer, Chrome, Firefox, and so on. The request goes to a server through the internet. In traditional systems, the server processes the request, renders the output, and sends the full response back to the client computer. The browser will then render the complete page and display the latest output.

Browsers process a page in a **DOM** (**Document Object Model**), also called an **HTML DOM**. Simply put, a DOM is a tree-like structure of all the HTML elements present on a page. It is an in-memory process, and stores all HTML elements in text form, in hierarchical order. A virtual DOM is an abstract copy of a DOM. It helps to first render changes to it and compare it with the actual DOM to identify what's changed.

Now, what React does in this situation is create a virtual DOM and only updates the elements on the page that gets new data. If any element does not get new data in response to a server request, it will remain unchanged and not be refreshed. Let's take an example of a client request. Let's assume you are on a website, filling out a **contact us** form. Once you click **submit**, the request will go to the server, and in return the server will respond with either a success or a failure message. Now, in this scenario, the traditional method is to update the whole page, which will take bandwidth and processing time. The other way is to keep the page as it is and only update the success or failure message. Which one do you think will be fast and responsive? The second one is the obvious answer.

Now that we have got some understanding of what React does and where it resides, let's move on to the next concept.

Node.js

Before Node.js, the JavaScript programming language had limited means of execution inside a browser environment. Node.js brings along functionality that breaks this restriction and opens up a new world of possibilities. It is a cross platform, open source, JavaScript runtime that lets users run JavaScript code outside of a browser environment. You can think of it as the ability to use JavaScript even at the server side rendering.

Node.js has a package manager called **Node Package Manager** (**NPM**) that contains thousands and thousands of open source components and libraries available to be used by any one.

One of the misconceptions is that React perforce requires Node.js to run. Node.js facilitates React development in different ways but it is not mandatory. Node.js helps to create a project structure and template to get started with a React Project. We will be using Node.js in the practical section of this chapter to build a React Application project structure using Node.js.

Similarly, Node.js provides a web server functionality to easily run your React application, and real-time change detection. In production or even in the development environment, you can run React applications on any other web server, such as Apache Tomcat, IIS, and so on; whichever you are comfortable with.

JSX

JSX is an extension to the JavaScript programming language. What it actually does is allow the use of HTML type syntax and tags within JavaScript, which makes it super easy to understand and program. It then, while compiling, converts this syntax into plain old JavaScript.

This extension to JavaScript actually requires a Babel framework to convert the JSX syntax to plain JavaScript syntax so that browser can understand. You can think of Babel as a JSX compiler. Without this compiler, the browser will not understand what your code actually means and will throw error exceptions.

Let's take an example of placing an HTML heading tag in a `<div>` element using JavaScript with React JS:

```
<div id="contentContainer"></div>

<script>
ReactDom.render(
            React.createElement('h1', null, 'Welcome'),
            document.getElementById('contentContainer'));
</script>
```

Now we will do the exact same thing, but this time, we will be using JSX syntax:

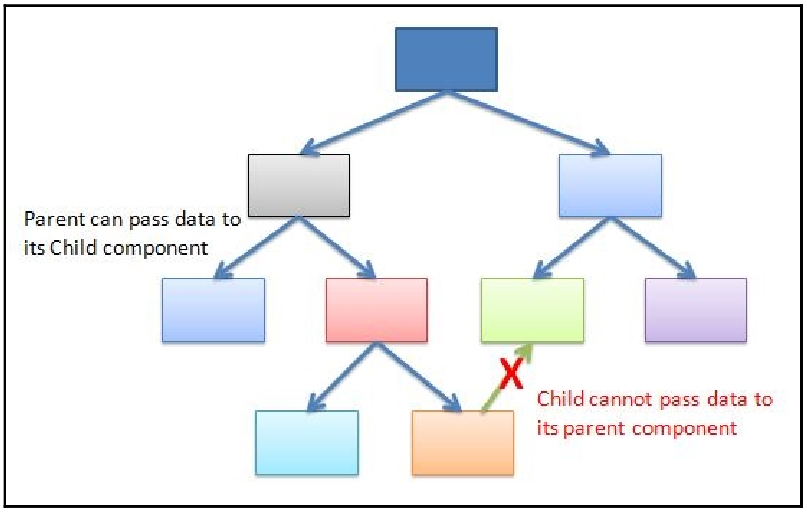
```
<div id="contentContainer"></div>

<script type="text/babel">
ReactDom.render(
            <h1>Welcome</h1>,
            document.getElementById('contentContainer'));
</script>
```

You may have noticed that we have added an attribute `type="text/babel"` to `<script>`. This is what tells the browser to execute the underneath code with the help of Babel compiler. This is just an example to give you a glimpse of how easy and readable our code becomes with the use of JSX. There are many different useful scenarios where JSX really makes writing JavaScript easy. You will see more JSX usage as we proceed in the chapter.

Unidirectional dataflow

In React, every element is a component. All the components are populated in such a way that they always make up a parent and child relationship. Now, when we talk about the direction of data flow, it is always unidirectional; it will always flow from a parent to its child.

It is not possible in any way for the child to be able to send data to its parent. Another important thing to note is that even the children cannot talk to each other. This means that the components cannot send data to the same or to the above level in the hierarchical concept. The following figure illustrates the data flow direction in a React application:


Parent can pass data to its Child component
X
Child cannot pass data to its parent component

Unidirectional dataflow in React

If your application architecturally requires your child components to pass on data to its parent components, then React is not suitable for your application.

Getting started with ReactJS

There are multiple ways to get you started. If you just want to see how React works, just create a single page application that calls all the required libraries from their React official CDN network to get you started so that you don't have to gather and download all the dependencies by yourself. To do actual development for a production environment, it is recommended to create a project with the help of NPM, which we will discuss as we proceed further. Let's first start by creating a single page React application.

Single page application

In its simplest form, you are required to include three JavaScript files in your project, which are as follows:

```
// 1 -- React Library
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>

// 2 -- React DOM Library
<script
src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

// 3 -- Babel for JSX compilation
<script
src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
```

The URL mentioned in this code is from the React CDN network; if you use it as it is then you won't have to download any files locally in your computer, but if at any time you disconnect from the internet, you will not be able to proceed with your development. The `helloworld.html` code is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Hello World</title>
    <script
src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script
```

```
src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script
src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
  </head>
  <body>
    <div id="root"></div>
    <script type="text/babel">

      ReactDOM.render(
        <h1>Hello, world!. Welcome to React App</h1>,
        document.getElementById('root')
      );

    </script>
  </body>
</html>
```

What this code does is link all the required js libraries from the React CDN, then creates a `div` element with `id` root that acts as the main content holder of your application. Then, using the React render function, a plain HTML heading containing your message gets returned and places it inside your content container. So here you go, your hello world single page app is ready.

Lets now proceed to build our React application professionally and in prescribed ways.

React application project

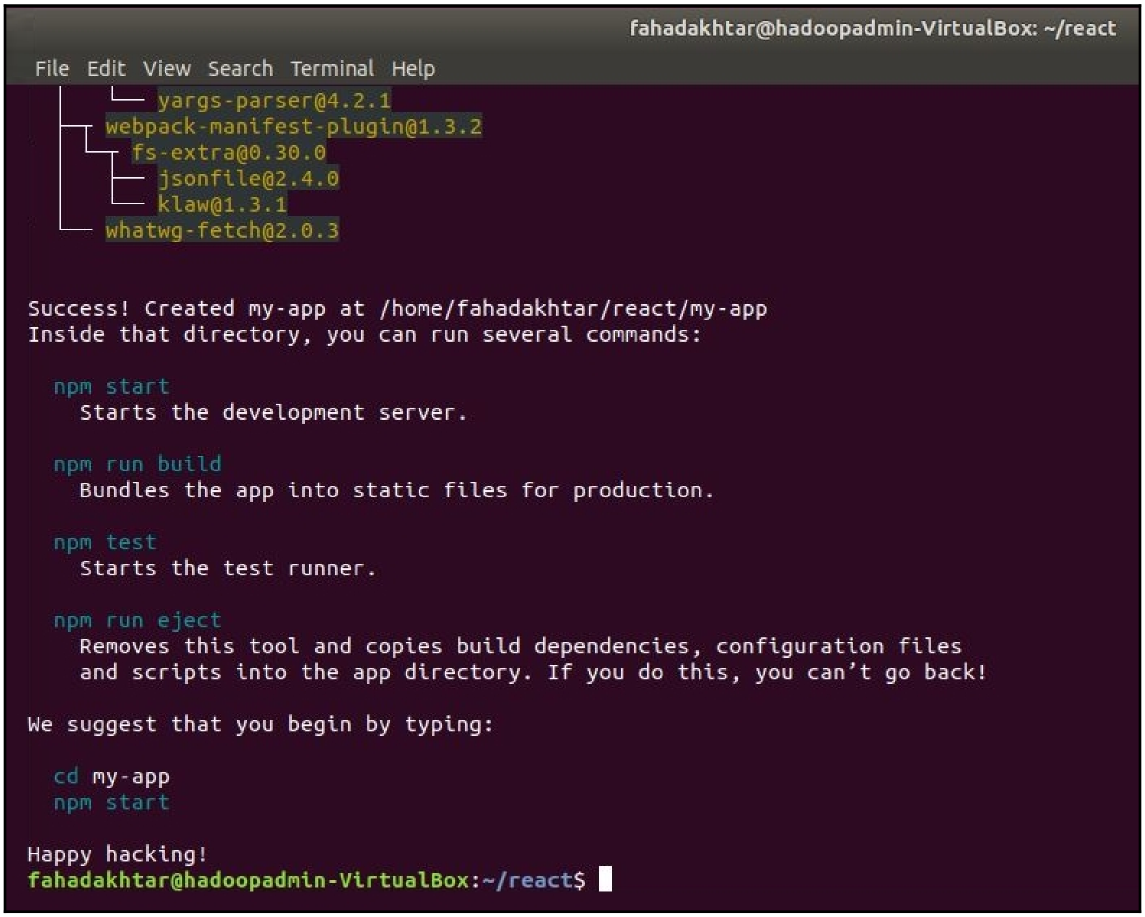
Now here comes the recommended way to create your React app project. It is recommended to create your project with the help of Node.js and its package manager. Using this method, you can build up the whole project structure in just a few steps. Let's start by installing Node.js and its components on our computer. This can be done by executing the following commands:

```
// This will install NodeJS framework
$ sudo apt-get install nodejs

// This will install NodeJS Package Manager (NPM)
$ sudo apt-get install npm

// This will install NPX. It is used to executes NPM binaries task
$ sudo npm install -g npx
```

Now that we have finished setting up the environment for Node.js, let's now create our React app using the predefined bundle. It comes with a preconfigured webpack, server, and babel so that you don't have to worry about any of the configurational stuff and can quickly start building your React app. To get started, execute the following command to set up your first React app:

```
// You can use any name instead of my-app. What ever name you mention
// it will create a folder with that name and set up your React app in it.
$ npx creat-react-app my-app
```

Once you execute this command, it takes couple of minutes to complete. You see something similar to the following screen at the end of the execution:


fahadakhtar@hadoopadmin-VirtualBox: ~/react
File Edit View Search Terminal Help

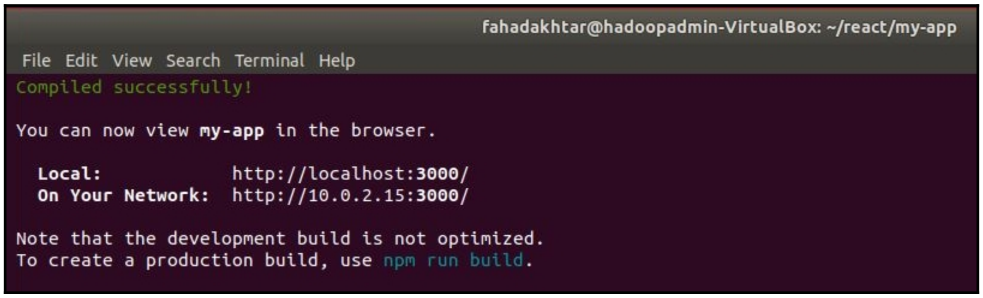
```
      └── yargs-parser@4.2.1
  ├── webpack-manifest-plugin@1.3.2
  │ └┬ fs-extra@0.30.0
  │   ├── jsonfile@2.4.0
  │   └── klaw@1.3.1
  └── whatwg-fetch@2.0.3


Success! Created my-app at /home/fahadakhtar/react/my-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd my-app
  npm start

Happy hacking!
fahadakhtar@hadoopadmin-VirtualBox:~/react$
```

NPX create React app completion

It suggests that you proceed with any of the steps mentioned as per your requirement. As this is the first time we have created our React app, we will begin by displaying the output as to what setup it has done for us. Let's execute the following command to start the server and execute the default app:

```
$ cd my-app
~/my-app$ npm start
```

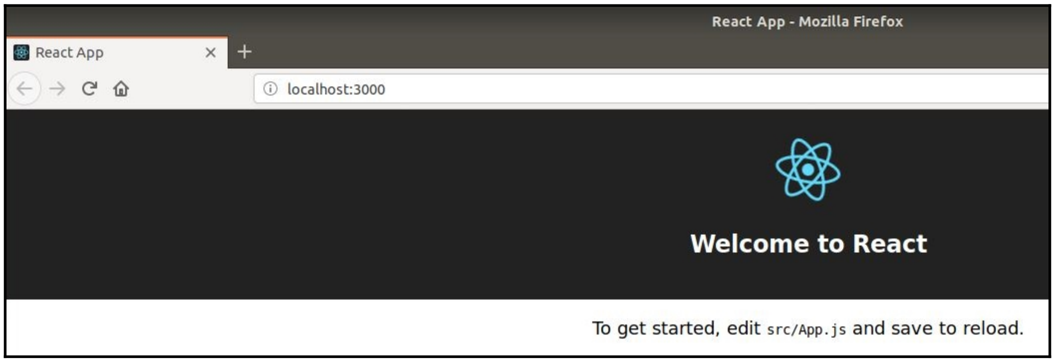
Once you execute the `npm start` command, it will compile your application, start a webserver, and publish it. The terminal window will show the following success message:


fahadakhtar@hadoopadmin-VirtualBox: ~/react/my-app
File Edit View Search Terminal Help

```
Compiled successfully!

You can now view my-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.0.2.15:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.
```

NPM Start terminal output

It will also automatically open up in your browser windows the URL mentioned in the terminal output window screen. If it will not open, you can manually open up a browser and point it to the URL mentioned. The default app installed with this project structure will appear similar to the following screenshot:



Welcome to React

To get started, edit `src/App.js` and save to reload.

Default React app

If you are able to see the default React app, it means that everything is set up correctly and we can proceed with modifying the default app and creating our own app. If not, please go back and redo the setup carefully.

Before proceeding to customize the app, let's first understand the directory and files created by NPX command for our React app.

React app directory structure

Let's have a look into the directory structure that is created by NPX. It contains the following directories and files:

```
my-app/
  README.md
  node_modules/
  package.json
  public/
    index.html
    favicon.ico
  src/
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
```

`node_modules/`: Contains the files and folders related to the dependencies while running our React App. It includes dependencies related to React JS, Babel, Webpack, and so on. We don't need to worry about the content in it as Node.js is responsible for managing it.

`public/`: Contains an HTML index page that actually contains the placeholder of the contents that we will populate using the React JS component. Make sure not to change the filename of `index.html` as it is the root placeholder of the components related to your app.

`src/`: Contains JavaScript files of all the components that we will use in our React App. Make sure not to change the name of `index.js` as it is the entry point of your application.

`package.json`: Node.js requires this file to contain the metadata of the application and all its related dependencies.

Now that we have understood the files and directories of our React app, let's now start customizing the existing components like `App.js` and also create our own components.

Components

In React, everything is a component. React applications are actually a collection of different components placed together. You can think of components like HTML tags. For example, in a web page, textbox is a component. Similarly, the submit button of the form is also a component.

Now if we broaden our thinking of components, a bigger component can contain many different smaller components. For example, a text editor on your webpage while writing an email; this would contain many different components, such as those that make your text bold, italic, underline, and so on. All these are small components inside the bigger component called text editor, which you place in your compose email webpage.

Similarly, while designing your React application, you have to think of each functionality as a component or child of a component. Let's take an example of creating a header component so that you can place it on all of your web pages easily. In order to create your component, the following code is required:

```
import React, { Component } from 'react';

class Header extends Component {
  render() {
    return (
      <h1>Welcome to my Website</h1>
    );
  }
}

export default Header;
```

This is just a very simple component that displays a welcome heading as an output. To call this component into your webpage is also super easy. In your `ReactDom` render function, simply call the variable as an HTML tag as shown in the following code:

```
ReactDom.render(<Header />, document.getElementById("headerHolder"));
```

This will execute the component code, get the returned HTML result, and place it in the `headerHolder` div component. As you can see, with the help of this component based approach code becomes easy to understand, and most importantly, it becomes reusable; wherever the component is required, simply call it by it tag and leave the rest to be handled by React and Babel.

When you try to return multiple parent elements, like a <h1> as well as an `<img>` tag in your component, it will not work. To make it work, wrap both the tags inside a `<div>` element, making it return a single parent element. Your code will become as follows:

```
import React, { Component } from 'react';

class Header extends Component {
  render() {
    return (<div>
                <h1>Welcome to my website</h1>
                <img src='logo.jpg' />
            </div>);
  }
}

export default Header;
```

Similarly, while rendering your component, if in any case you require calling it multiple times, wrap it as many times as you want but inside a div element. Your code will become as follows:

```
ReactDom.render(<div>
                    <Header /><Header /><Header />
                </div>, document.getElementById("headerHolder"));
```

Let's now move on to making our component more complex and generic by introducing the concept of properties.

Properties

Properties are also referred to as React Props. They are used to pass values from the React render function to the components. They are similar to the attributes of any HTML tags. Here, it's up to you to name these attributes whatever you want. They are very useful for making generic templates and reusable code. For example, let's say that your task is to create a component that displays an image with its name and year of posting. Imagine; without properties, you will be required to create this component for each and every image. But with the help of properties, you will create a generic template of how the image and description will appear on the page. Then, while calling the component, you can pass on the required details as an attribute for rendering.

Attributes can be accessed inside the component using `this.props.propertyName`. Let's now take a quick review of the following code to display an image with title and posting year:

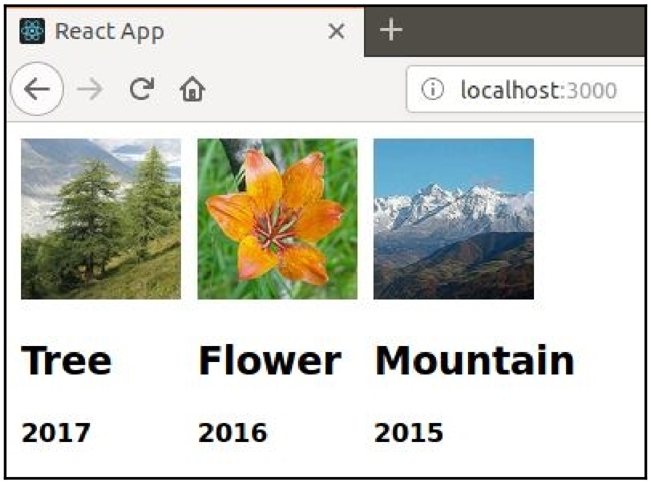
```
import React, { Component } from 'react';

class Image extends Component {
  render() {
    return (
      <div>
          <img src={this.props.file} />
          <h2>{this.props.title}</h2>
          <h4>{this.props.postyear}</h4>
      </div>
    );
  }
}

export default Image;
```

Now that we have a generic template for an image component, let's now call it for our images. Now our `index.js` file has the following code:

```
import React from 'react';
import ReactDOM from 'react-dom';
import Image from './Image';
import './index.css';
import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(
  <div className='imgdiv'>
    <Image file='tree.jpg' title='Tree' postyear='2017' />
    <Image file='flower.jpg' title='Flower' postyear='2016' />
```

```
        <Image file='mont.jpg' title='Mountain' postyear='2015' />
      </div>
    , document.getElementById('root'));

    registerServiceWorker();
```

If you now look at the output in the browser by pointing it to the local web server URL, it will be similar to the following screenshot:



Output of custom React Image component

Now that you have a good idea about how beneficial these properties are and how to use them, let's now move on to the next topic, event handling.

Event handling

In some cases, we require our components to respond when the end user interacts with them; for example, a submit button on a form. When the user clicks on it, it is supposed to take all the data filled by the user and send it to the server for whatever further reason it is meant for. Similarly, you can define these interactions, or better, do event handling within React on the client side.

Let's expand our example of an image gallery. Now, suppose you want to display some information related to an image whenever the user clicks on the picture. To handle an event, first of all you have to define a function, similar to the render function, which will handle that specific event. It will be similar to the following inside your component class:

```
description = () => {
    alert('Some description related to image');
}
```

This will display a message window to the user with the text as mentioned. But our component is not specific to a single image, so we will use a React property to pass on the description along with other attributes. Now, our code will become as follows:

```
description = () => {
    alert({this.props.children});
}
```

`this.props.children` is used to get the text written in between the HTML tags without needing to define an attribute. We will now set our image to call this function whenever a user clicks on it using the `Javascript OnClick` function. It will be done using the following code:

```
<img onClick={this.description} src={this.props.file} />
```

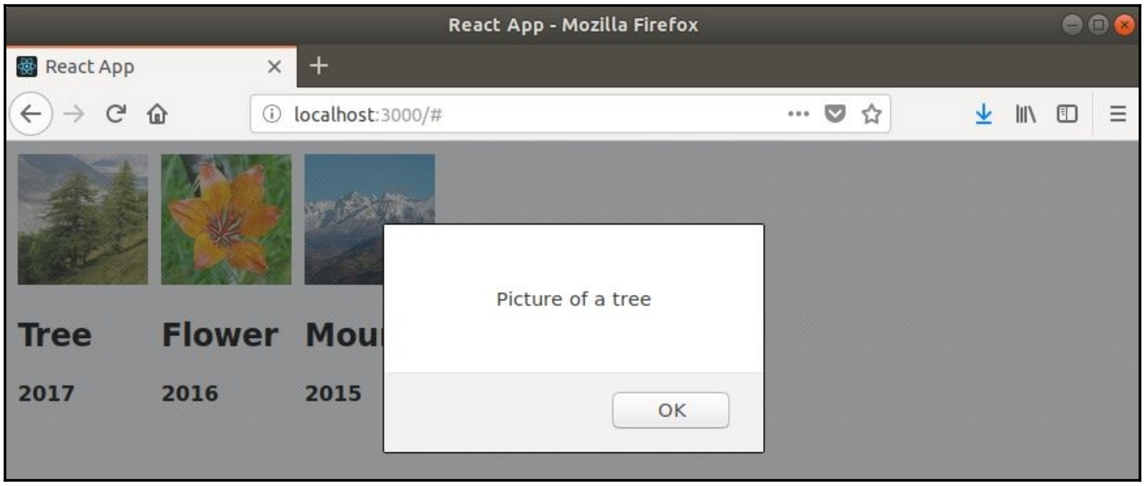
The complete code for the `Image.js` component class file is as follows:

```
import React, { Component } from 'react';

class Image extends Component {
  description = () => {
      alert(this.props.children);
  }

  render() {
    return (
      <div className='imgdiv'>
           <img onClick={this.description} src={this.props.file}
className='gallery' />
          <h2>{this.props.title}</h2>
          <h4>{this.props.postyear}</h4>
      </div>
    );
  }
}

export default Image;
```

The complete code for `index.js` is as follows:

```
import React from 'react';
import ReactDOM from 'react-dom';
import Image from './Image';
import './index.css';

ReactDOM.render(
  <div className='imgdiv'>
    <Image file='tree.jpg' title='Tree' postyear='2017'>Picture of a
tree</Image>
    <Image file='flower.jpg' title='Flower' postyear='2016'>Picture of a
Flower</Image>
    <Image file='mont.jpg' title='Mountain' postyear='2015'>Picture of a
Mountain</Image>
  </div>
, document.getElementById('root'));
```

When the user now clicks on any image, it will display the short description mentioned in between the `image` tags. The following figure shows the output when the user clicks on the tree image:



Picture of a tree

Event handling on image click

Let's now move on to our next topic of managing the state of elements.

State

One of the characteristics of React Props is that once you set it for any component, you cannot change it for its entire lifecycle. Now, if you need to set a property of a component and want the flexibility of also changing it, you need to utilize the State object in React.

To understand the concept of state properly, let's take an example of a list of colors, populated in a dropdown HTML element, with which a user can select a desired color according to his/her preferences. As an initial stage, we will set a default color, which will also become the background color of the text on the webpage. Once the user selects a different color, this color will immediately change.

Let's start by initializing the state object. We will initialize a color variable in the state object and assign it a default color. This is done using the following code:

```
constructor(props) {
     super(props);
     this.state = { color: 'Yellow' };
 }
```

An important thing to remember is that the state initialization has to be done in the `constructor` function of your component class.

Now we will write our event handler function, which will update the state object once the user selects a different color. In the previous section, we have already covered how to handle events. So, the code of the event handler function in this scenario is as follows:

```
 setColor = (event) => {
     this.setState({color: event.target.value});
 }
```

Now that we have initialized the `state` object and have our event handler ready, let's build up a list of colors using the HTML select element and assign our `setColor` event handler function on the `onChange` event of the element. The component `render` function code is as follows:

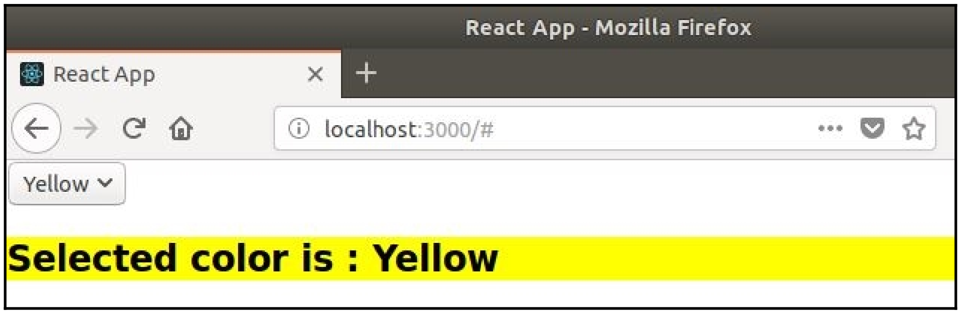
```
render() {
   var selectedColor = this.state.color;
    return (
      <div>
         <select onChange={this.setColor} value={selectedColor}>
          <option>Red</option>
          <option>Green</option>
```

```
              <option>Yellow</option>
              <option>Blue</option>
            </select>
            <br/>
            <h2 className={selectedColor}>
              Selected color is : {selectedColor}
            </h2>
      </div>
        );
      }
```

Here I have defined four css classes so that, upon the changing of the color, the specific css class will be assigned. Copy the following into your `index.css` file:

```
    .Red {
      background-color: Red
    }
    .Green {
      background-color: Green
    }
    .Yellow {
      background-color: Yellow
    }
    .Blue {
      background-color: Blue
    }
```

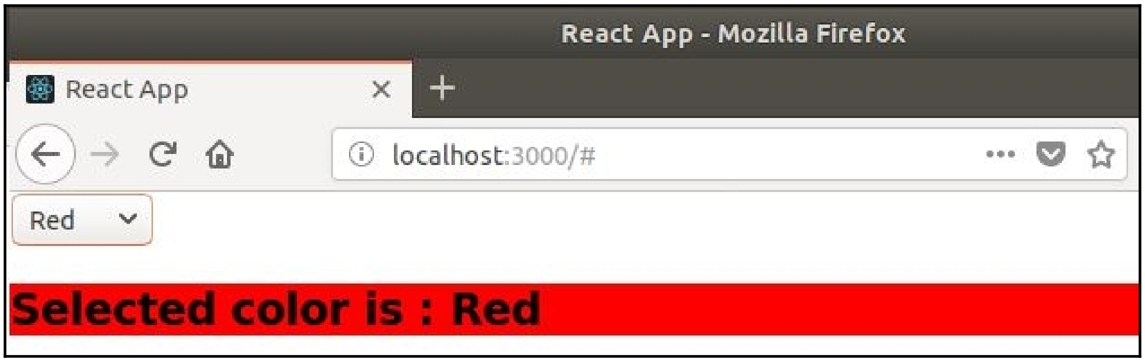
Now open your browser and point it to your local web server to see the output. The following figure shows the initial results:



Selected color is : Yellow

Initial output of the dropdown component

Now let's assume that a user has selected the color `Red` from the dropdown menu. Once a new color gets selected, it will update the state, which results in an update to the text as well as the background color. The following screenshot show the result upon selection:



Selected color is : Red

Output as a result of user selection

In order to give you a clear picture, the complete `Dropdown.js` file code is as follows:

```
import React from 'react';

class Dropdown extends React.Component {
  constructor(props) {
      super(props);
      this.state = { color: 'Yellow' };
  }

  setColor = (event) => {
      this.setState({color: event.target.value});
  }

  render() {
   var selectedColor = this.state.color;
    return (
      <div>
         <select onChange={this.setColor} value={selectedColor}>
          <option>Red</option>
          <option>Green</option>
          <option>Yellow</option>
          <option>Blue</option>
        </select>
        <br/>
        <h2 className={selectedColor}>
             Selected color is : {selectedColor}
        </h2>
        </div>
      );
    }
  }

export default Dropdown;
```

And the `index.js` code is as follows:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import Dropdown from './Dropdown';

ReactDOM.render(<Dropdown />, document.getElementById('root'));
```

We have now got a clear understanding of how state works and how it is different from properties. In the previous example, we have also covered event based state changes; whenever our desired event happens, the state value will be updated, and, based on the state, our webpage elements will be updated by React.

The complete code of all the React components developed in the preceding examples can be download from the GitHub repository using this link: `https://github.com/smfahad/BDAH-React-Examples.git`.

Let's now move on to our next topic, which is also related to state management, but which uses a different library altogether, named Redux. It will also cover state management from a different prospect and in a broader sense.

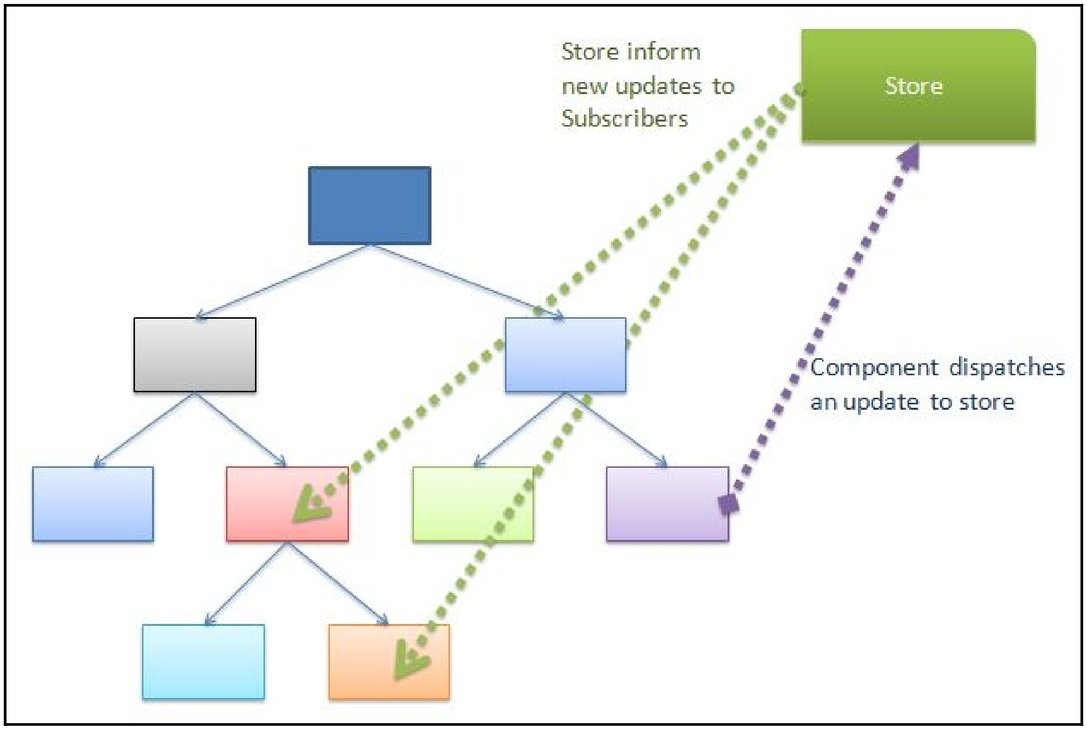
Redux

In React, data flows in a unidirectional manner, which we covered in an earlier topic in this chapter. Now, if you get a scenario where a child component has to notify its parent or any other component regarding its state change, what will happen in that scenario? React does not support data flow from a child component to its parent's component, not even to other components.

Here Redux library comes into the picture. It provides a way for all the components in the application to talk to each other. Let's now explore the architecture of Redux.

Architecture of Redux

Redux comes up with a solution to React's unidirectional data flow. The solution is that all the components should maintain their states in a global place, which will be called the **Store**. The responsibility of the store is to get updates from the components. Components send updated data to the store, which is also called **Dispatch**, and the components that need to know about such changes should register with the store to receive updates - this is called **Subscribe**. This will allow components to send and receive updates across the application without having to worry about unidirectional data flow. The following figure illustrates the architecture of Redux:


Store inform new updates to Subscribers
Store
Component dispatches an update to store

Redux architecture

A point to notice here is that components are not sending updates directly to other components. They are first sending it to the store, and the store will then notify subscribers regarding new changes.

Key concepts

Following are the key concepts related to Redux:

- Single store
- Action
- Reducer

You should have a clear understanding of both the architecture of Redux and these key concepts before diving into Redux practically. The following sections cover these concepts.

Single store

It is recommended to use single store architecture in your application. It is also known as single place of truth. All the components dispatch and subscribe to a single store that can therefore hold the application's state intact.

Action

You cannot change a state unless an action happens. This means that the state will be in read-only mode unless any intended user actions have been performed. It is also referred to as a payload that contains the data that will be sent to the store. It will only specify that something happened, but what will be done as a result of that is not defined by Action.

Reducers

Reducer is a function that contains the logic of how the state will change as a result of an Action. It takes the previous state and an Action to emit a new state as a state change. It is also known as pure functions.

Now that we have a basic understanding of Redux architecture and the key concepts, let's dive into the practical stuff.

Guestbook application

In this example, we are going to create a guestbook application. Users can come and view other users' comments and add their own by providing the name and the desired comments.

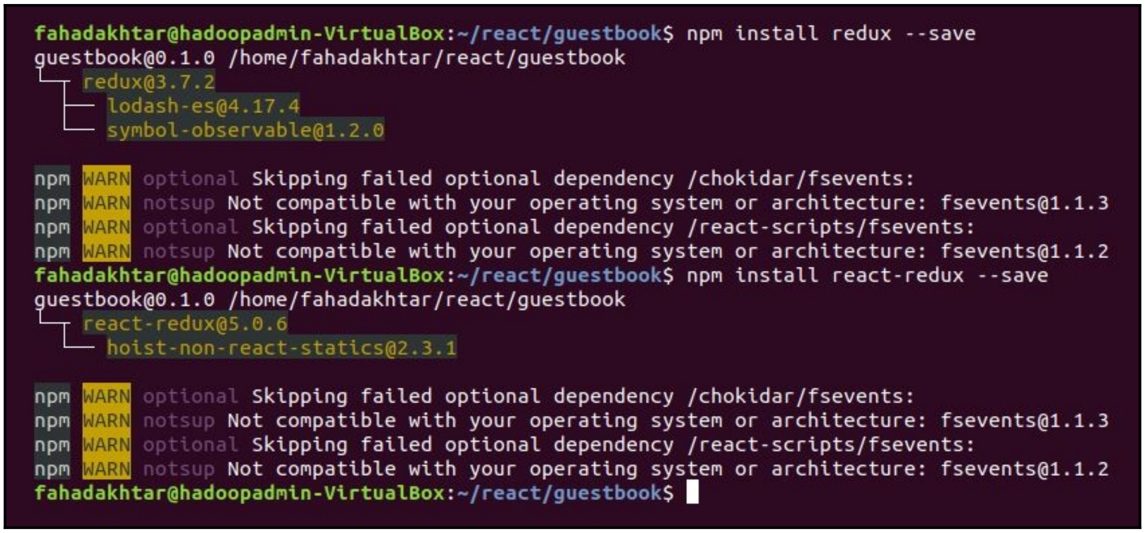
In the first step, we will create a new React application named **guestbook**, using the same procedure mentioned earlier in this chapter. We will then add Redux and React-Redux libraries in it as follows.

Installation

Now that we have created our new project, it's time to add the libraries required for getting started with Redux. To do so, you need to run a Node package command to download the required libraries into your project, and save the dependency description into your package.json file. Execute the following command to get started with Redux:

```
$ npm install redux --save
$ npm install react-redux --save
```

The following figure illustrates the output of the `install` commands:



```
fahadakhtar@hadoopadmin-VirtualBox:~/react/guestbook$ npm install redux --save
guestbook@0.1.0 /home/fahadakhtar/react/guestbook
└─┬ redux@3.7.2
  ├── lodash-es@4.17.4
  └── symbol-observable@1.2.0

npm WARN optional Skipping failed optional dependency /chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.1.3
npm WARN optional Skipping failed optional dependency /react-scripts/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.1.2
fahadakhtar@hadoopadmin-VirtualBox:~/react/guestbook$ npm install react-redux --save
guestbook@0.1.0 /home/fahadakhtar/react/guestbook
└─┬ react-redux@5.0.6
  └── hoist-non-react-statics@2.3.1

npm WARN optional Skipping failed optional dependency /chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.1.3
npm WARN optional Skipping failed optional dependency /react-scripts/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.1.2
fahadakhtar@hadoopadmin-VirtualBox:~/react/guestbook$
```

React Redux installation

Make sure to execute the install commands inside your application folder.

Create a store

In order to create a store, Redux and React-Redux libraries provide the functions `createStore` and `Provider`. These functions need to be imported into your application. We will import them to our `index.js` file using the following code:

```
import { createStore } from "redux";
import { Provider } from "react-redux";
```

Once imported, you need to create a store. To do so, use the following code:

```
const store = createStore(reducer);
```

Here you may have noticed that the `createStore` function takes reducer as a parameter. We have already discussed what Reducers are and their purpose, so let's now understand how to set up a reducer in our application.

Setting up Reducer

Reducer is actually a function, which takes the current state (some may call it the last state) and the action performed. Reducers only act as a result of an action, and change the current state to a new state with the updated values. To set up a reducer, execute the following code:

```
const initialState = {
  commentList: []
};

function reducer(state = initialState, action) {
    switch (action.type) {
        case "AddComment":
              return {
                commentList: state.commentList.concat(action.payload)
              };
        default:
          return state
  }
}
```

First of all, we have initialized State here, so that if the current state is undefined it will not throw an exception. Inside the reducer function, using the action type, we have defined what the new state will be. In this case, as a result of an `AddComment` action, it will add the newly added comment to the existing list and return it to the store. You can add as many functions as you want inside Reducer. We will discuss the **action** parameter in the *Setting up Dispatcher* section.

Now that we have set up our reducer, it's now time for the `AddComment` component to dispatch the new changes, so that it will be updated in the store.

Setting up Dispatcher

In our case, we have set up an `addComment` function, which will be triggered as a result of a submit button click. In this function, we will dispatch comments using the `props.dispatch` function. The code for this is as follows:

```
addComment(event) {
    const comment = {
      name: this.state.name,
      comment : this.state.comment
    }
    this.props.dispatch( { type: "AddComment", payload: comment});
}
```

The must have argument for a dispatch function is `type`, which will tell reducers what action is to be taken. `payload` is just a variable to pass on the new values to the store. You can name it anything you want, like data, newValues, and so on. It will be available in your reducer function's `action` parameter.

Connect function

The most important part of using Redux code is establishing a connection between the component and the Redux store. This can be done by importing the connect function from the React-Redux library at the beginning of your component:

```
import { connect } from 'react-redux';
```

Add the `connect` function call at the end of your component file, within your component `export` declaration:

```
export default connect()(AddComment);
```

The complete code of `AddComment.js` is as follows:

```
import React, { Component } from 'react';
import { connect } from 'react-redux';

class AddComment extends Component {

  onCommentUpdate(field, event) {
  if (field === 'name') {
    this.setState ({name: event.target.value});
  } else if (field === 'comment') {
    this.setState ({comment: event.target.value});
  }
  }

  addComment(event) {
      const comment = {
    name: this.state.name,
    comment : this.state.comment
  }
  this.props.dispatch( { type: "AddComment", payload: comment});

  }

  render() {
    return (
  <div style={{padding:10}}>
    <h2>Guestbook</h2>
    <input type="text" onChange={this.onCommentUpdate.bind(this, 'name')}
placeholder="name" />
    <br/>
    <textarea type="text" onChange={this.onCommentUpdate.bind(this,
'comment')} placeholder="Add your comments" />
    <br/>
    <button onClick={this.addComment.bind(this)}>Add Comments</button>
  </div>
    );
  }
}

export default connect()(AddComment);
```

Now that we are successfully able to send the updates to the Redux store, it's time to set up subscribers so that they can receive new updates.

Setting up Subscribers

The components that are intended to receive updates regarding any changes to the state of a component will receive changes using props. In our case, we have a component called `DisplayComment`, which will list all the comments available in the Redux store.

In our index.js file, while calling the component, the following is required:

```
    <Provider store={store}>
        <DisplayComment commentList={this.commentList} />
    </Provider>
```

As you can see, we are passing `commentList` as a property to the `DisplayComment` component. Inside the components, it will be used like any other props that we did in our React tutorial. But the point to remember here is that we also need to connect this component to the Redux store. This will also be done using the `connect` function. After importing the `connection` function, we have to set up a function to transform state values into properties. This will be done at the end of your display component, and the code for this purpose is as follows:

```
const mapStateToProps = state => ({
  commentList: state.commentList
});

export default connect(mapStateToProps)(DisplayComment);
```

The following is the complete code of the `DisplayComment.js` file:

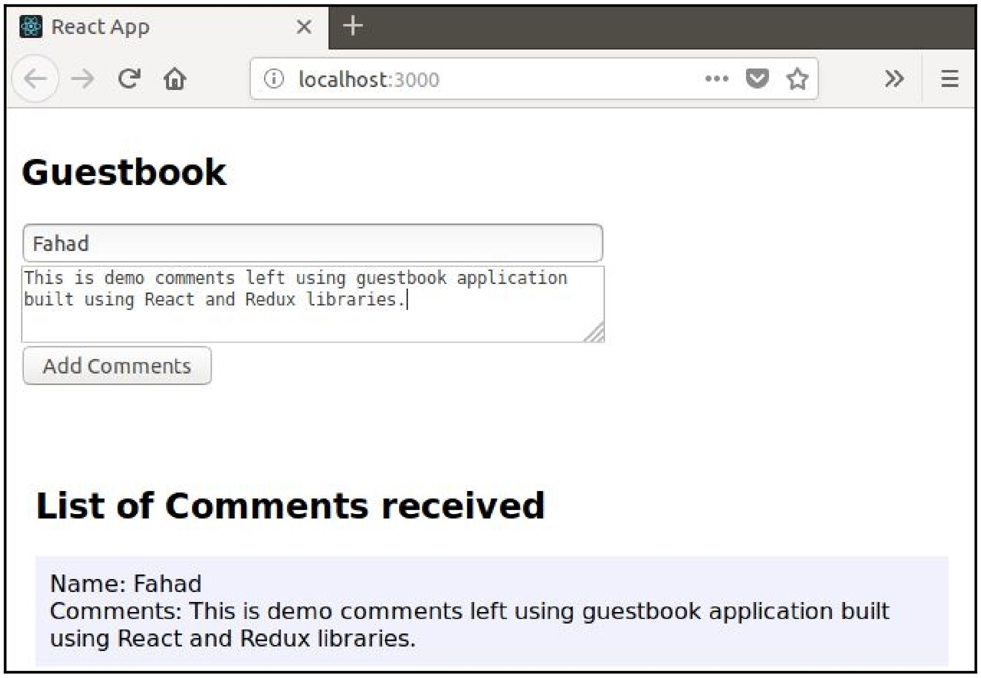
```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import Comment from "./Comment"
class DisplayComment extends Component {

  render() {
    return (
      <div style={{padding:20}}>
        <h2>List of Comments received</h2>
        {this.props.commentList.map((comments) =>
          <Comment name={comments.name} comment={comments.comment} /> )}
      </div>
    );
  }
}

const mapStateToProps = state => ({
  commentList: state.commentList
});

export default connect(mapStateToProps)(DisplayComment);
```

The complete code of the `index.js` file is as follows:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import AddComment from './AddComment';
import DisplayComment from './DisplayComment';
import { createStore } from "redux";
import { Provider } from "react-redux";

const initialState = {
  commentList: []
};

function reducer(state = initialState, action) {
  switch (action.type) {
    case "AddComment":
      return {
        commentList: state.commentList.concat(action.payload)
      };
    default:
      return state
  }
}

const store = createStore(reducer);

ReactDOM.render(
  <div>
    <Provider store={store}>
      <AddComment />
    </Provider>
    <br/>
    <Provider store={store}>
      <DisplayComment commentList={this.commentList} />
    </Provider>
  </div>
  , document.getElementById('root'));
```

Final output

After putting all our effort into setting up a React and Redux based application, we will get our application ready and similar to the following screenshot. It can be enhanced in many ways, but as far as understanding React and Redux is concern, this does the job:



Guestbook

Fahad

This is demo comments left using guestbook application built using React and Redux libraries.

List of Comments received

Name: Fahad
Comments: This is demo comments left using guestbook application built using React and Redux libraries.

Redux based React application final output

This guestbook application is just to give you a start towards creating a more complex, interactive, and responsive application. React library is very useful for creating responsive and interactive websites, and with Redux's help, it overcomes the limits that it had.

The complete code of the Guestbook application can be downloaded from the Github repository using this link. `https://github.com/smfahad/BDAH-React-Redux-Guestbook.git`.

Summary

In this chapter, we have learned about React and its different components. We have learned about its architectural design and key concepts such as JSX, unidirectional data flow, and so on. We then dived into the practical part. Firstly, we created a single page React application, and then later we moved in a more professional direction by creating a React project using Node.js package manager. We then went through different key areas with respect to development, such as components, properties, state, event handling, and so on.

We then moved into another library, Redux, which resolved the limitations of React's unidirectional data flow. We went through its architecture and different key concepts to give you a strong conceptual hold. We then moved into its coding part. We started with how to set up a Redux application on top of your React application. Then, moving step by step, we covered the key areas of Redux library, such as creating a Store, setting up Reducers and Actions, how to Dispatch an update, connecting components to Redux store, and how Subscriber will receive updates using a practical example.

In next chapter, we will learn about different technologies for designing scalable and distributed backend Architecture.

11
Backend Architecture

Designing a good backend architecture for any application has its advantages. It helps in analyzing the data, increase productivity time, increase performs, and most of all, it will allow you to better handle different scenarios of your application. If the case is vice versa, then your application may not be at its best in terms of performance and it may not help you analyze its data, which may also lead to bad customer experience.

Nowadays every business is somehow connected to another business and exchanging information. With the increased demand in online shopping and online social activities, it is very important for business to open its gates to share information with a third party, which in this technology era is through exposing its software to online world. There are multiple ways of doing it that we will also be exploring in this chapter.

Then comes a turning point, which is the performance of an application. It is a very important factor for any application's success or failure. It can drive a customer in and out very easily. All other factors may then seem secondary. In this chapter, we will also explore how to increase application performance.

In this chapter, we will mainly be covering the following:

- RESTful API
- Different types of request methods
- Different types of authentication mechanisms
- Redis
- Installation and setting up Redis server and client
- Different working scenarios and operations

Now, in your development life, you may have heard the term API. Let's find out what it is all about and why it is important.

API

API stands for **application programming interface**. It is used to allow a third-party application to interact with the desired application having predefined methods or calls. Each method is exposed to receive a call with or without some parameters and as a result, it gives a response with related output.

There are many different types of APIs available. Some of the examples are computer APIs, desktop APIs, different type of sensors APIs, in short if any hardware or software is open to allow any third-party application to interact with them, it has to expose an API for that purpose.

Here we will mainly be exploring web-based API. There are two types of web-based API. The first one is **Simple Object Access Protocol** (**SOAP**) and the other one is **Representational State Transfer** (**REST**). Basically, they are different in terms of technology. SOAP is a protocol for web service. It means that it has some specific guidelines to exchange structured information using XML documents, whereas REST is an architectural design that will interact with other applications through HTTP requests. Both type of APIs are programming language independent, which means you can use both types of API in your preferred programming language.

Nowadays, the most common and preferred way of communication over the web is through REST. So now we will be exploring REST API, which is also known as RESTful API.

RESTful API

REST stands for **Representational State Transfer**. It is also referred to as RESTful API. It is an architectural style of exposing a web service. It services a request mainly over HTTP using GET, POST, PUT, and delete delivery methods. We will discuss these **Hypertext Transfer Protocol** (**HTTP**) methods in the subsequent section.

RESTful web service gets a request using a HTTP URL. Other related information such as authorization data will be passed on along with its header. Then it's up to the REST service provider to return results in whichever format suits them, but it is recommended to provide a response of a REST API call in a **JavaScript Object Notation** (**JSON**) object. Almost all the publicly available REST API follow the same standards.

We will now explore different types of request methods that we can utilize while making a HTTP request.

HTTP request methods

As discussed earlier, there are mainly four methods using which REST API receives a request. These are as follows:

GET

As the name states, this method is used to pull a request from the required server. A typical example of a GET request is when you click on any link on a website or point your browser to a specific address.

POST

It is used to send user generated data to the server. A typical example of a POST request is request generated as a result of submitting a form. It can be for the purpose of create or update.

PUT

It is also used to create or update data on the server. The main difference between PUT and POST is that the PUT request will replace the actual data. As a result, the created or updated data will always be same. Whereas a POST request will create or update the data incrementally or as per the defined backed logic.

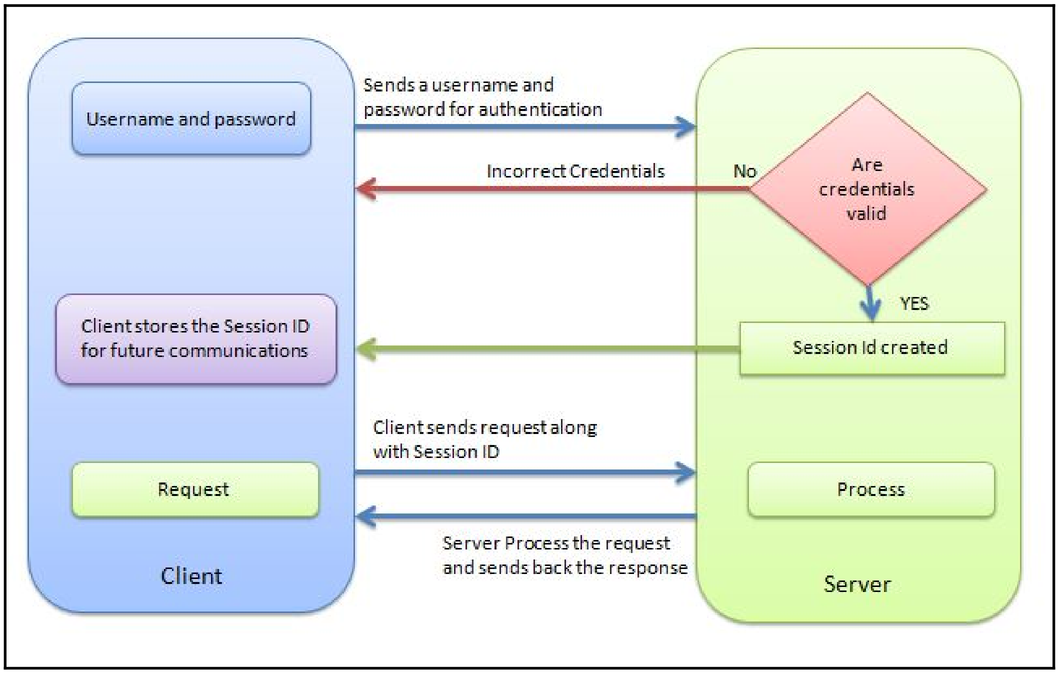
DELETE

It is used specifically to delete the data on the server using an HTTP request.

Now let's suppose you have a REST API available on the web to be utilized by other applications, but it will be limited to those who have a valid license or a purchase agreement or similar thing. In simple words, you don't want unauthorized user to access your services. Let's now explore how you can implement authentication in your REST API.

Authentication

Let's first have an overview of how authentication works in a traditional system. It initializes by sending a username and password by client to server. Once received, the server validates whether the provided credentials are correct or wrong. If it's wrong, the server sends an incorrect message back to the client to provide a valid login. If it's correct, the server creates a session and sends the session ID back to the client for future communication. So from next time, instead of providing username and password, the client will send the request along with the session ID to identify itself as an authorized user. The following diagram illustrates a traditional authentication system:


Username and password
Sends a username and password for authentication
Incorrect Credentials
No
Are credentials valid
YES
Client stores the Session ID for future communications
Session Id created
Client sends request along with Session ID
Request
Process
Server Process the request and sends back the response
Client
Server

Traditional client server authentication mechanism

This type of client server communication is also referred to as stateful communication. Whereas REST API doesn't maintain or store any session state or client side state and that's why it is also called **stateless API**. It means that every time a client sends a request to server, it somehow has to pass either username and password or any other mode of authentication along with the request to identify itself as an authorized user:

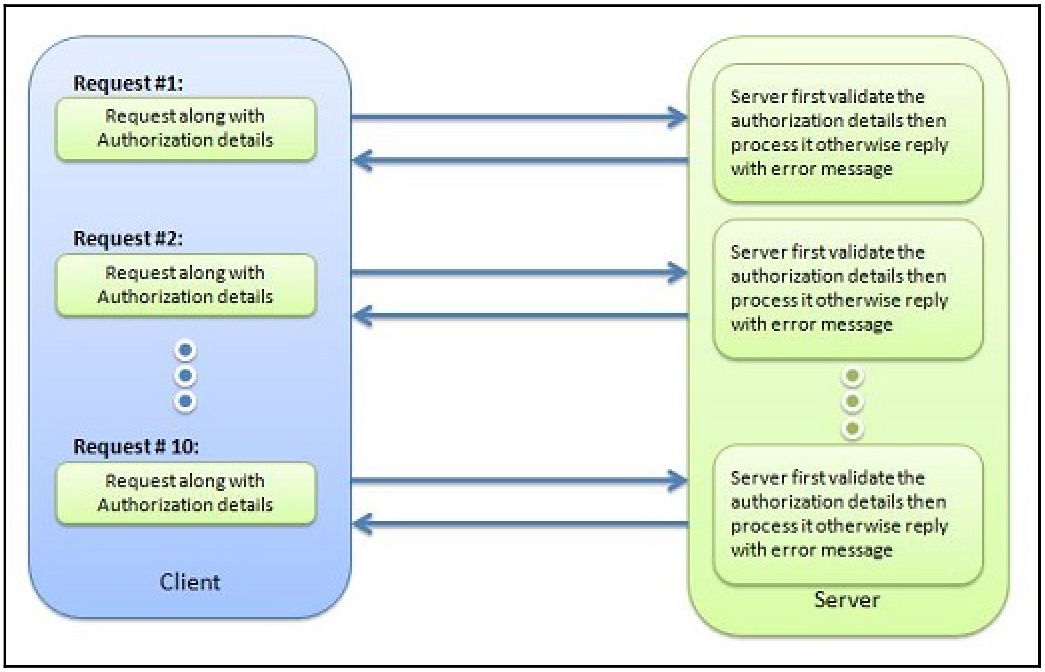
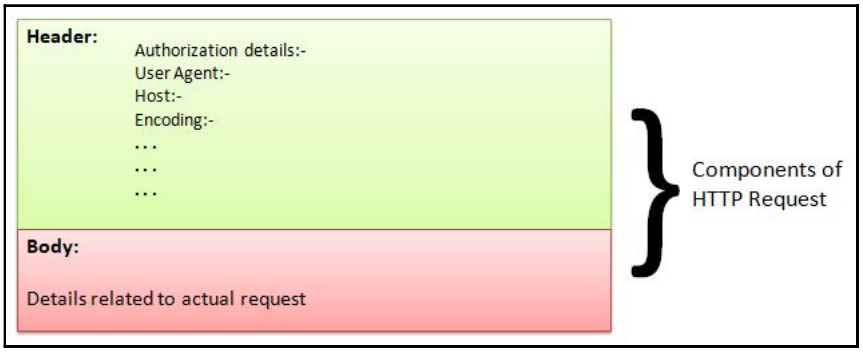

Request #1:
Request along with Authorization details
Request #2:
Request along with Authorization details
Request # 10:
Request along with Authorization details
Client
Server first validate the authorization details then process it otherwise reply with error message
Server first validate the authorization details then process it otherwise reply with error message
Server first validate the authorization details then process it otherwise reply with error message
Server

Figure: Client-server communication using REST web service

HTTP request packet can broadly be categorized in two sections, header and body. Header contains client-related information such as request method, user agent, authorization details, host information, and so on, and body contains the actual request. The following diagram illustrates a HTTP Request:



Components in HTTP request

REST API supports different types of authentication mechanism. Some of them are as follows:

- Basic authentication (basic auth)
- JSON web tokens
- OAuth
- OAuth 2
- Digest access authentication
- Asymmetric cryptography

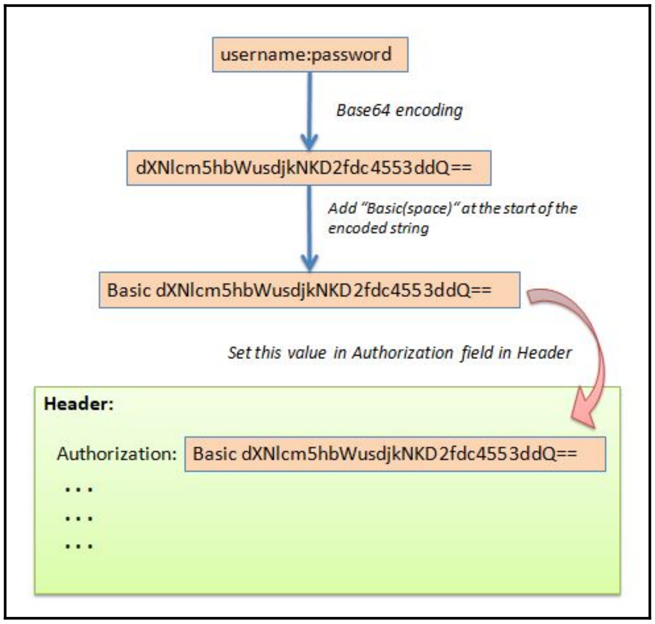
We will discuss the first two authentication mechanisms in detail, which are basic auth and JSON Web Token in the subsequent section. Let's start with the first one, basic auth.

Basic authentication

In HTTP header, we have a field called authorization. In basic authentication mechanisms, we pass on the username and password details in the header authorization field. The steps that we have to follow are as follows:

1. First concatenate the username and password, separated by a colon (`:`) sign. For example, `username:password`.
2. Encode the concatenated credentials using `Base64` encoding.
3. Again concatenate the encoded string with the word `"Basic(space)"` in the start.
4. Add this final string in your header authorization string.

The following diagram illustrates these steps in graphical form:


username:password
Base64 encoding
dXNlcm5hbWusdjkNKD2fdc4553ddQ==
Add "Basic(space)" at the start of the encoded string
Basic dXNlcm5hbWusdjkNKD2fdc4553ddQ==
Set this value in Authorization field in Header
Header:
Authorization: Basic dXNlcm5hbWusdjkNKD2fdc4553ddQ==
...
...
...

Steps to set up basic authentication parameters in request headers

An important point to notice here is that `Base64` is an encoding mechanism, it is not an encryption mechanism. So, if someone gets holds to the encoded information, it can easily be decoded using the `Base64` decode function.

Now when the server receives the request, it will check the authorization field in the header. If it finds the word `Basic`, it will help to understand the server that the authorization mechanism used here is basic auth. It will then follow the same steps in reverse order to get the username and password, which will be as follows:

1. The server will extract the authorization field value and match whether the first word contains `Basic`.
2. Once it finds the word `Basic`, it will split the string into two using space as a splitting end.
3. It will then decode the second part using `Base64` decoding.
4. It will then split the username and password using the colon (`:`) as split end.
5. It will then validate that the username and password is correct. If yes, it will service the request, otherwise it will return with an error message.

This is how a complete cycle works in basic authentication mechanisms. Remember, this communication is not secure. If anyone gets hold of the header information of the request, username and password information can easily be extracted from it. To make it secure, always send your request over Hypertext Transfer Protocol Secure (HTTPS), which will encrypt the complete request.

Always use HTTPS to send requests that contain critical information such as username and password, credit card info, and so on. It will encrypt the complete request and become almost impossible to get attack.

You may be thinking that if Base64 is not securing the information then why we are doing the encoding and decoding. The reason being is to convert the username and password into HTTP compatible characters, so that if either field contains any non compatible character, this process will convert them to HTTP compatible.

Let's now move on to the next authorization mechanism used in REST API, which is JSON Web Token.

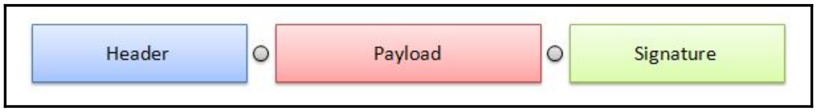
JSON Web Token

Similar to basic authentication, JSON Web Token also sends the authentication details in the HTTP Header authorization field. The difference is that JSON has its own format to pass on these details.

JSON Web Token contains three parts separated by dot (.):

1. Header.
2. Payload.
3. Signature.

The following diagram illustrates the JSON Web Token structure:



JSON web token structure

Let's now explore each part of the token.

Header

It mainly contains two pieces of information in JSON format. The first part is type, which will be **JWT** (JSON Web Token) and the second one is the hashing algorithm, which will be used to encrypt the signature part. The format will be as follows:

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

While concatenating the header part to create a JSON Web token, it should be encoded using Base64 algorithm.

Payload

It contains the actual data in the form of JSON that is required to be transmitted. The following is an example of a decoded payload:

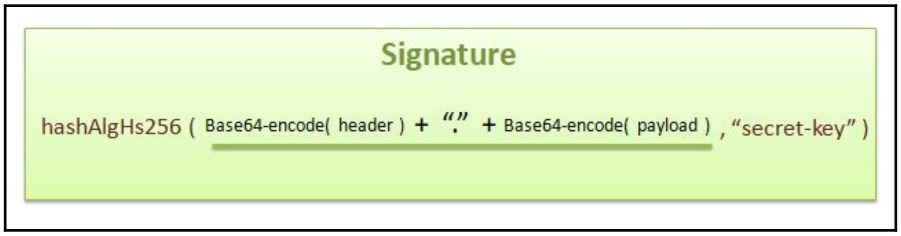
```
{
  "tokenID": "1234567890",
  "username": "John Doe",
  "role": "admin"
}
```

Similar to the header information, before adding it to JWT, it is also required to be encoded using the Base64 algorithm.

Signature

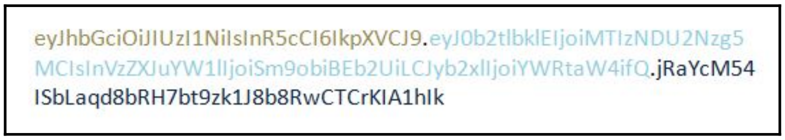
It is the main component of **JWT**. It will let the server validate the complete token to confirm that it was originated from a trusted source. Its content contains Base64 encoded header concatenated with dot (.) and then again concatenated with Base64 of payload data. This will then get encrypted using the hashing algorithm with a secret-key. There are many different hashing algorithms such as HS256, RS256, and so on.

The following diagram illustrates the contents of the Signature part in JWT using the HS256 function:


Signature
hashAlgHs256 (Base64-encode(header) + "." + Base64-encode(payload) , "secret-key")

Structure of signature in JSON Web Token

The final output of JSON Web token will be similar to something as follows, which will be added to the Authorization field along with HTTP Header. The following diagram illustrates the final format of JWT:



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbklEIjoiMTIzNDU2Nzg5
MCIsInVzZXJuYW1lIjoiSm9obiBEb2UiLCJyb2xlIjoiYWRtaW4ifQ.jRaYcM54
ISbLaqd8bRH7bt9zk1J8b8RwCTCrKIA1hIk
```

JSON web token final output

You can visit the JWT website `https://jwt.io/`, which provides a tool to experiment and play around with a different look and feel of JWT.

Let's now move on to practical stuff and build a RESTful web service and a client to consume it using Java.

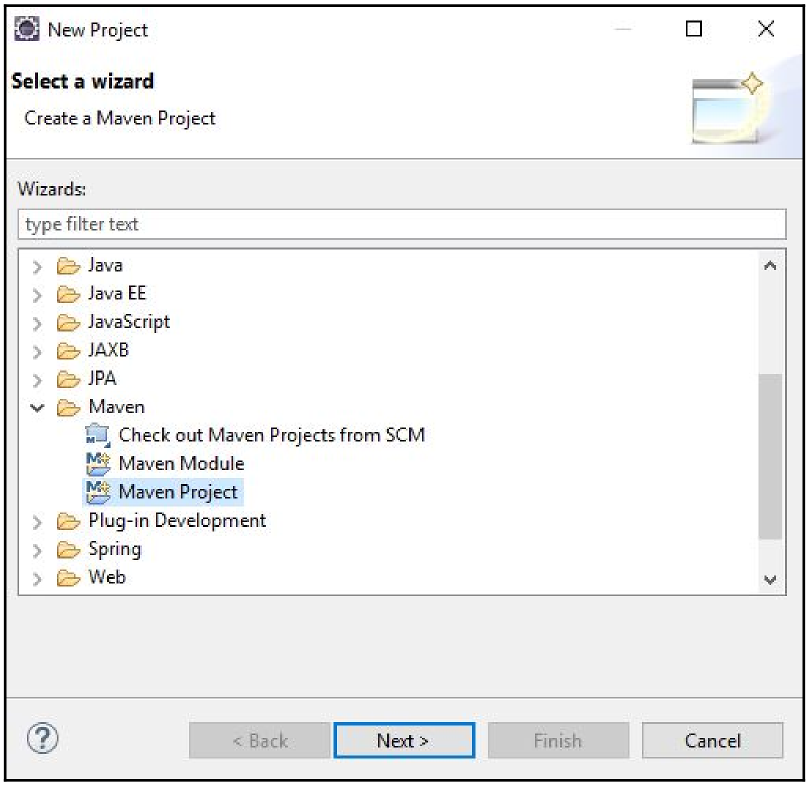
Practical

We now have understanding of REST API and how it works. Let's now create our own service and then later we will create a client for it to consume. For this purpose, we will be using Java as a programming language and Eclipse IDE. Let's first start by creating a RESTful web service.

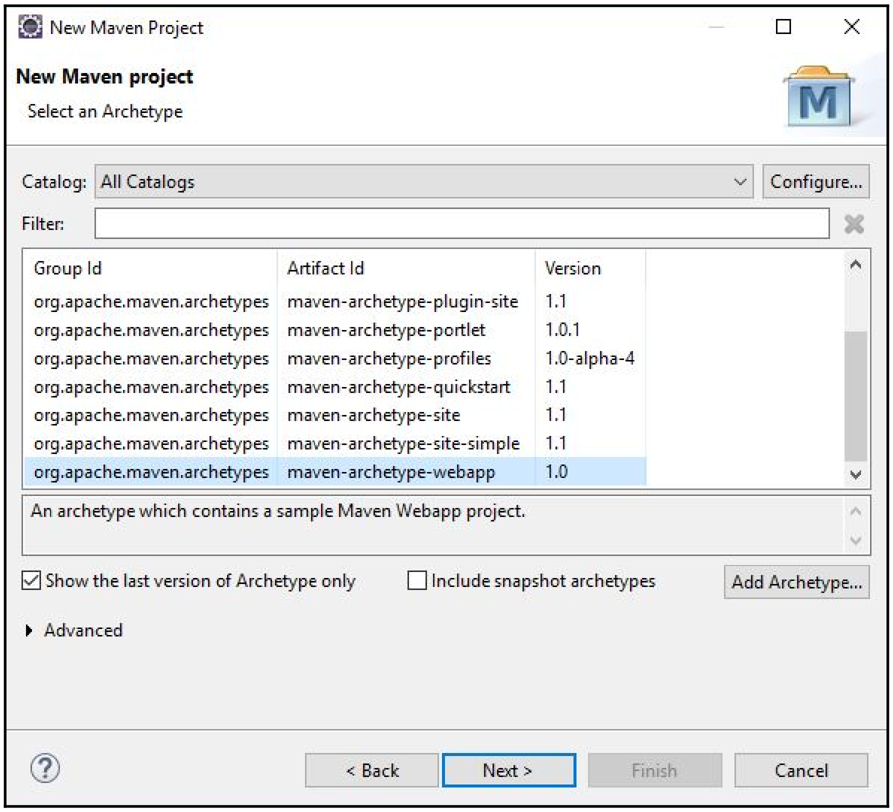
RESTful web service

Follow this step-by-step guide to create a RESTful web service:

1. Open Eclipse, create a new `Maven Project`, and then click `Next`:



2. On the **Archetype** selection screen, select `maven-archetype-webapp` from the available list:


New Maven Project

New Maven project

Select an Archetype

Catalog: All Catalogs

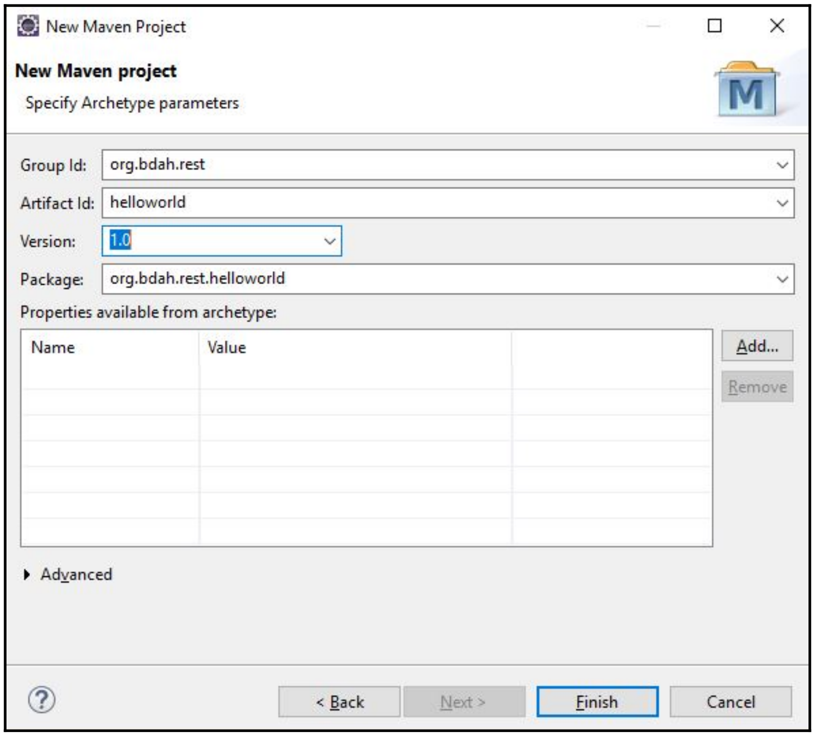
Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

An archetype which contains a sample Maven Webapp project.

☑ Show the last version of Archetype only ☐ Include snapshot archetypes

3. Fill out the parameters on the **Archetype** parameter screen, which in our case is as follows:



Group Id: org.bdah.rest
Artifact Id: helloworld
Version: 1.0
Package: org.bdah.rest.helloworld

4. Click `Finish` to create the project.

It's now time to import some dependencies required to build our REST API. For this purpose, open the `pom.xml` file and copy the following code in between the `<dependencies></dependencies>` tag:

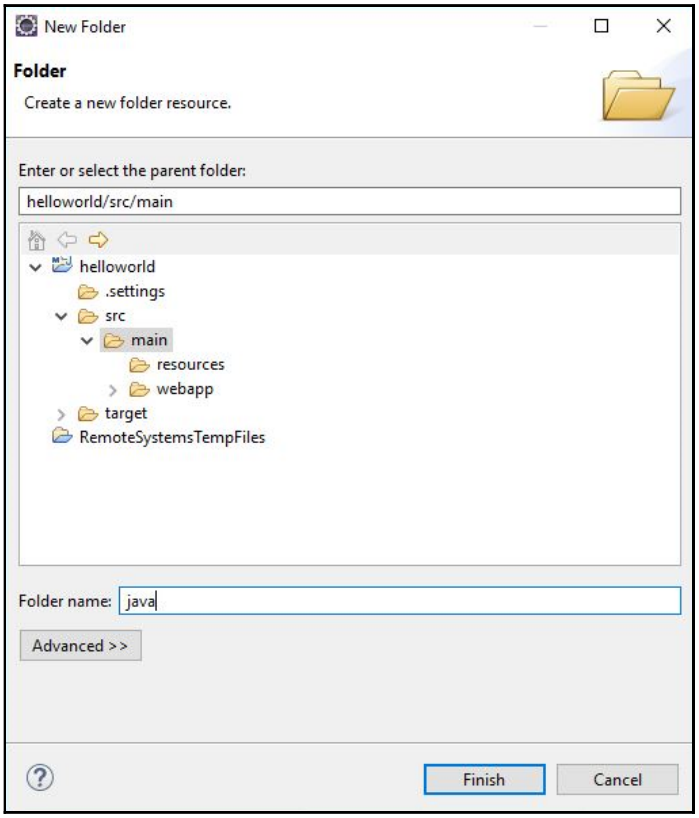
```
<dependency>
    <groupId>org.glassfish.jersey.bundles</groupId>
    <artifactId>jaxrs-ri</artifactId>
    <version>2.26</version>
</dependency>
```

Save the `pom.xml` file and wait for a minute or two. Eclipse will automatically download the related jar files.

The next step is to create Java classes. But before that, first create a folder in your project where you will keep all the class files. By default, the Maven project doesn't create it for you, so you have to do it by yourself. Follow these steps for this purpose:

1. Right-click on the project and select **New**.
2. A new window will pop up, select **folder** and click **Next**.
3. Select the project you are working on, drill down in it until the hierarchy tree reaches to `src/main`, and then select it. In the `Folder name:` box, write the name `java` and click `Finish`. It will create the required folder.

The following diagram illustrate these steps:



Java folder creation inside src/main

Now create a new class called `HelloAPI` inside the `org.bdah.rest.helloworld` package. The following code is self explanatory as each line contains comments to explain the code. Copy the following code in it to create your first REST API:

```
package org.bdah.rest.helloworld;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

// It will the endpoint of your REST API
@Path("/hello")
public class HelloAPI {

  // It will define the action at which the following method will
  // called upon
  @GET
  // It will define that the following method produces
  // result in HTML format
  @Produces(MediaType.TEXT_HTML)
  public String helloMsg() {
    // Once called upon, the REST API will return the following message
    return "<h1>Hello World. Message from REST API</h1>";
  }
}
```

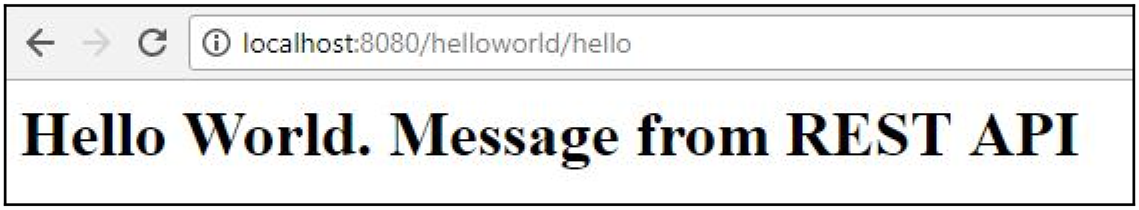
The next step is to create a new class called `MyRestApplication` and copy the following code in it. It is required to make the REST API project to start up and publish as a service:

```
package org.bdah.rest.helloworld;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

// It define the base path of the published REST API
@ApplicationPath("/")
public class MyRestApplication extends Application {

}
```

Now for the final step. We will now configure the tomcat server where you will publish the newly developed REST API locally by following this step-by-step guide:

1. Click on **File** | **New** | **Other**.
2. Look for **Server**, select it, and click **Next**.
3. Scroll down and select **Tomcat v8.0 Server** from the list and click **Add**.
4. Click on the **Download and Install** button to automatically download Tomcat binary and configure the path. Then click **Finish**.
5. Click **Finish** again to finish setting up the server.

Now that we are all set up and ready to go, it's now time to start the application. Right-click on your project and select **Run as** | **Run on Server**. Now point your browser to the published URL, as shown in the following screenshot:



Hello World. Message from REST API

Locally published REST API service

If you are able to see the screen similar to the preceding screenshot, it means you have successfully created your first RESTful web service. If you are seeing something else, please revisit the REST Web service step-by-step guide to check out the issue.

A point to notice here is that if we closely look in to the URL, `localhost:8080` is the server where we have published our REST API, `helloworld` is the project name that we have created, and `hello` is the path that we have configured in our `HelloAPI` Java class as an entry point for the outside world to get response.

The complete project code of the RESTful Web Service is available to download at `https://github.com/smfahad/BDAH-RESTful-Webservice-Basic.git`.

In the next section, we will create a Java program that will act as a client to consume this service.

Java client

Follow this step-by-step guide to create a RESTful web service:

1. Open Eclipse and create a new **Maven Project**, then click **Next**.
2. On the **Archetype** selection screen, select **maven-archetype-quickstart** from the available list.
3. Fill out the parameters on the **Archetype** parameter screen, which in our case is as follows:
    - **Groupid**: `org.bdah.rest`
    - **Artifact id**: `restClient`
    - **Version**: `1.0`
4. Click **Finish** to create the project.

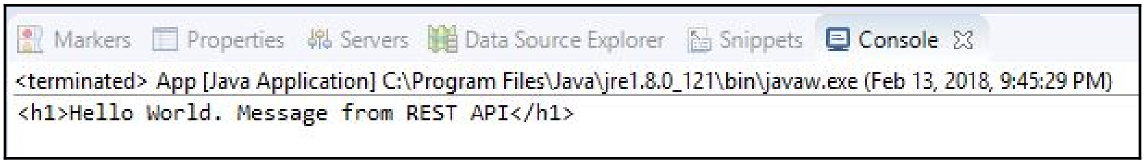
It's now time to import some dependencies required to build our REST API. For this purpose, open the `pom.xml` file and copy the following code in between the `<dependencies></dependencies>` tag:

```
<dependency>
    <groupId>org.glassfish.jersey.bundles</groupId>
    <artifactId>jaxrs-ri</artifactId>
    <version>2.26</version>
</dependency>
```

Save the `pom.xml` file and wait for a minute or two. Eclipse will automatically download the related JAR files. Now open up the `App.java` class file and write down the following code in it:

```
package org.bdah.rest.restClient;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;

public class App
{
    public static void main( String[] args )
    {
      // Initialize a client object through which we
      // will request REST API
      Client client = ClientBuilder.newClient();
      // Assigning a URL which REST web service to call
      WebTarget target =
client.target("http://localhost:8080/helloworld/hello");
      // Printing the output in console
      System.out.print(
          // Request for the response
          // Return type to be HTML
          // and the object type as String
          target.request(MediaType.TEXT_HTML).get(String.class)
          );
    }
}
```

It's now time to start the application. Right-click on your project and select `Run as` | `Java Application`. You will see similar output as shown in the following screenshot:



```
<terminated> App [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 13, 2018, 9:45:29 PM)
<h1>Hello World. Message from REST API</h1>
```

Output of Java REST API client

If you are able to see output similar to the preceding screenshot, it means your Java client is programmed correctly. If not, please revisit the Java client section from the start and try to create the project again.

Complete project code of the Java client is available in GitHub repository at `https://github.com/smfahad/BDAH-REST-Java-Client.git`.

We will now move our discussion to in memory databases, which are mostly used to cache and improve overall application response. It is called Redis.

Redis

Redis is an in-memory data structure store, released under open source BSD license. It is also referred to as Remote (Re) Dictionary (di) Server (s). It works on the principle of Key Value pair. Redis supports many different types of datasets such as lists, maps, sets, sorted sets, and so on.

It's in memory data that can also be made persisted on disk. It means that if, because of any reason, a Redis instance got rebooted, or if after recovering from a failure it comes back online, then it will restore all of its data from the disk and place it in memory as if nothing happened. This is what makes Redis utilized as a database.

It is super fast in performance because it runs in memory, due to which many people are using Redis for cache purpose in order to increase performance of their applications.

Redis has a publisher and subscriber module, which is also referred to as the Pub/Sub module. In this module, Publisher creates channels and the subscriber joins them. Now, whenever a publisher sends a message to its channel, all the subscribers will automatically receive the message without having to manually check for it after a defined interval.

Let's now move on to the practicals and start with the installation of Redis.

Installation

The current version of Redis is 4.0.8. Open up the Terminal and execute the following commands in order to install Redis:

```
// Create Redis folder inside home directory and opened it
$ mkdir redis
$ cd redis

// It will now download the current version of Redis
// To check out the latest version, please visit http://redis.io
$ wget http://download.redis.io/releases/redis-4.0.8.tar.gz

// This command will extract the downloaded package into redis folder
$ tar xzf redis-4.0.8.tar.gz
$ cd redis-4.0.8

// This will build Redis
$ make
```

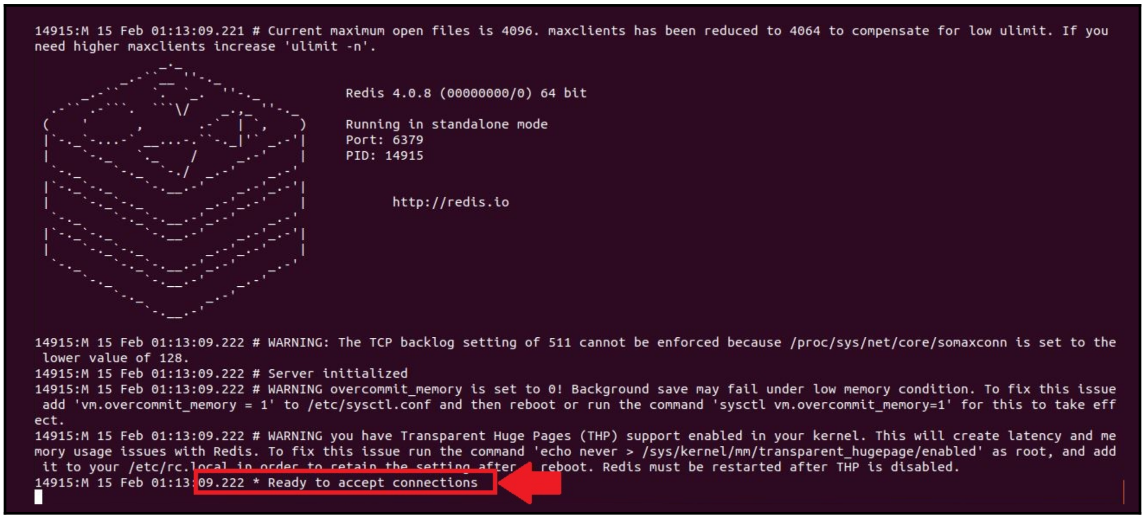
Now that we have downloaded and built Redis, it's time to start the Redis server.

Redis server

In order to start the Redis server, execute the following command:

```
$ src/redis-server
```

Remember to execute this command inside the folder resulted in extraction of the downloaded file. In our case, it is `~/redis/redis-4.0.8/`. Once you execute this command, it will display a screen similar to the following screenshot:



```
14915:M 15 Feb 01:13:09.221 # Current maximum open files is 4096. maxclients has been reduced to 4064 to compensate for low ulimit. If you
need higher maxclients increase 'ulimit -n'.
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 4.0.8 (00000000/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 14915
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

14915:M 15 Feb 01:13:09.222 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the
 lower value of 128.
14915:M 15 Feb 01:13:09.222 # Server initialized
14915:M 15 Feb 01:13:09.222 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue
 add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take eff
ect.
14915:M 15 Feb 01:13:09.222 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and me
mory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add
 it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
14915:M 15 Feb 01:13:09.222 * Ready to accept connections
```

Redis server started

If you see the last message, which is `Ready to accept connections`, as highlighted in the preceding screenshot, it means that your Redis server has been started successfully and listening to the port for new connections.

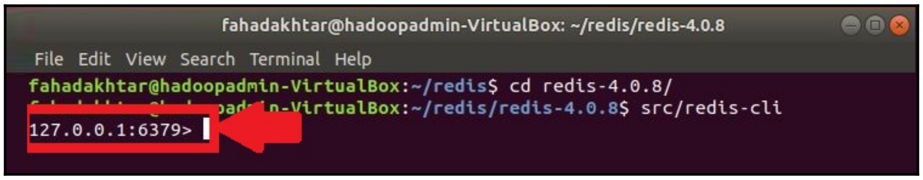
Keep the Terminal window open in which you start Redis server. It will keep it alive. Once you close it, it will stop accepting connections.

Redis client

To start Redis client, execute the following command:

```
$ src/redis-cli
```

Remember to execute this command inside the `Redis` folder. In our case, it is `~/redis/redis-4.0.8/`. Once you execute this command, it will display a screen similar to the following screenshot:


fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help

```
fahadakhtar@hadoopadmin-VirtualBox:~/redis$ cd redis-4.0.8/
fahadakhtar@hadoopadmin-VirtualBox:~/redis/redis-4.0.8$ src/redis-cli
127.0.0.1:6379>
```

Redis client

If you see the `127.0.0.1:6379>` prompt after executing the command, it means you have successfully connected to Redis server and are ready to perform and execute Redis commands.

Let's now move on to learn some Redis-related commands that you will be using a lot when working with Redis.

Working with Redis

All the Redis-related commands are required to be executed inside Redis client. So make sure to always open the `redis-cli` client before the execution of the commands.

Redis data types and structures

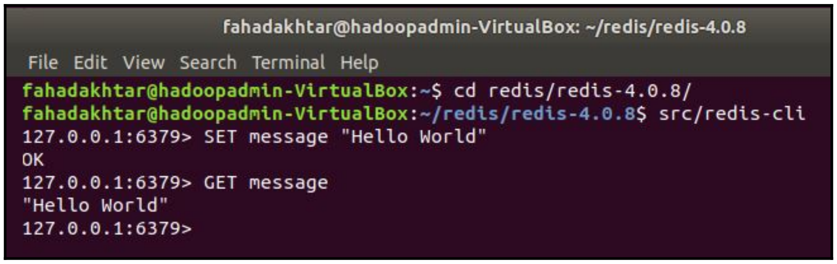
There are different types of data types and data structures that Redis supports, which includes string, HashMap, list, sets, sorted sets, and so on. Let's start with the basic one.

String

It is very much similar to defining any variable in any programming language. By using the `SET` and `GET` methods, you can provide a string that will become the key and then provide a string to store the value against the key, in case of setting a value. Execute the following commands in `redis-cli` to `SET` and `GET` keys and values:

```
// Following command is used to set a key with its value
127.0.0.1:6379> SET message "Hello World"

// Following command is used to retrieve the value of a key
127.0.0.1:6379> GET message
```

The following screenshot illustrates the execution and output of these commands:


fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help

```
fahadakhtar@hadoopadmin-VirtualBox:~$ cd redis/redis-4.0.8/
fahadakhtar@hadoopadmin-VirtualBox:~/redis/redis-4.0.8$ src/redis-cli
127.0.0.1:6379> SET message "Hello World"
OK
127.0.0.1:6379> GET message
"Hello World"
127.0.0.1:6379>
```

Illustration of SET and GET for string data type

HashMap

In Redis, HashMap stores values in combination of key value pairs. It supports the following commands to store and retrieve data using Redis HashMap.

The following command is used to create a `credentials` HashMap with keys `username` and `password` and values against them:

```
127.0.0.1:6379> HMSET credentials username fahad password akhtar
```

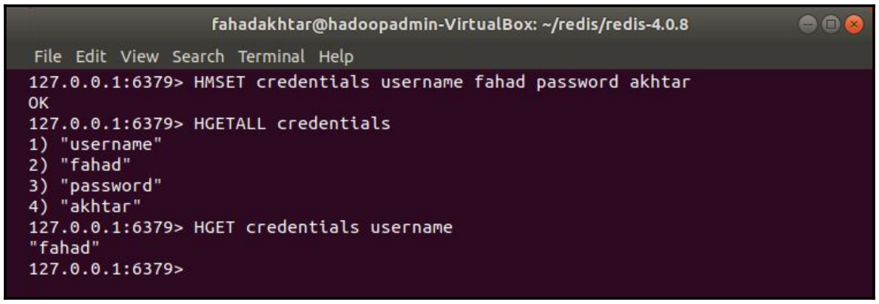
The following command is used to extract all the key values available in the `credentials` HashMap:

```
127.0.0.1:6379> HGETALL credentials
```

The following command is used to extract a specific key `username` from the `credentials` HashMap:

```
127.0.0.1:6379> HGET credentials username
```

The following screenshot illustrates the outputs of these commands:

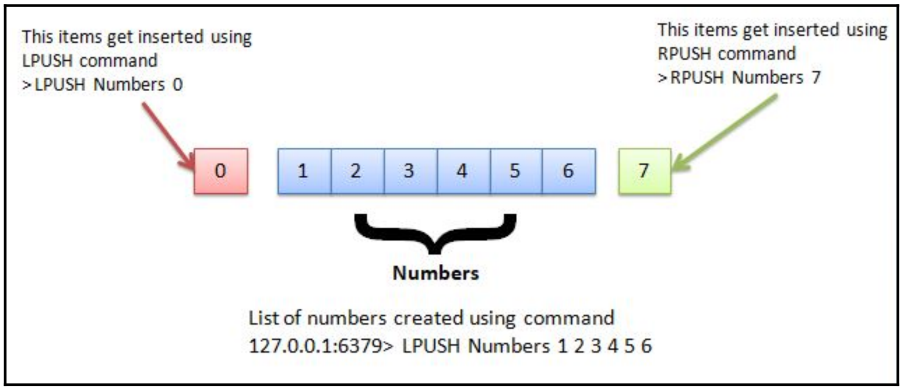

fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help

```
127.0.0.1:6379> HMSET credentials username fahad password akhtar
OK
127.0.0.1:6379> HGETALL credentials
1) "username"
2) "fahad"
3) "password"
4) "akhtar"
127.0.0.1:6379> HGET credentials username
"fahad"
127.0.0.1:6379>
```

Redis HashMap commands and outputs

List

It is like a normal list that contains the different items in it. It is sorted as the items get inserted in it using `LPUSH` and `RPUSH` commands. If you insert items using the `LPUSH` command, it will be inserted at the beginning of the list and if you use `RPUSH`, it will be added at the end of the list. The following diagram illustrates the structure of a list:


This items get inserted using
LPUSH command
>LPUSH Numbers 0
This items get inserted using
RPUSH command
>RPUSH Numbers 7
0
1
2
3
4
5
6
7
Numbers
List of numbers created using command
127.0.0.1:6379> LPUSH Numbers 1 2 3 4 5 6

Redis list structure of LPUSH and RPUSH commands

The following commands are used to insert values into a list:

```
// New list created with these values
127.0.0.1:6379> LPUSH Numbers 1 2 3 4 5 6

// New item inserted at the beginning of the list
127.0.0.1:6379> LPUSH Numbers 0

// New Item inserted at the end of the list
127.0.0.1:6379> RPUSH Numbers 7
```

Similarly to delete an item from the list, use the following commands:

```
//To remove an item from the start of the list
127.0.0.1:6379> LPOP Numbers

//To remove an item from the end of the list
127.0.0.1:6379> RPOP Numbers
```

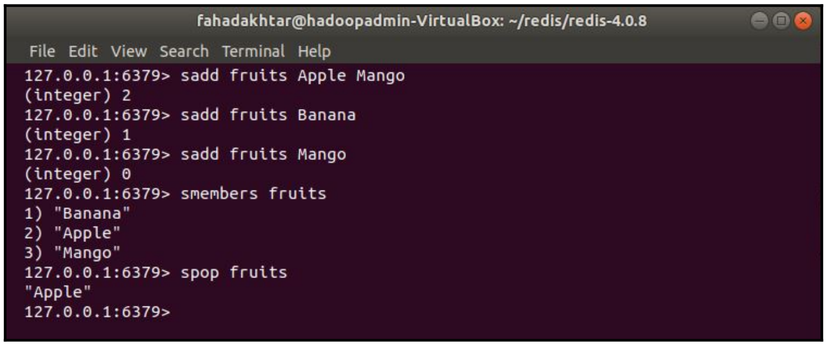
Set

It is a collection of unordered items, but only unique items. It doesn't store duplicate values. If your write is successful in the set, it will return `(integer) 1`, but if you try to insert duplicate values, it will return with `(integer) 0`.

The following commands are used to manipulate `Sets` in Redis:

```
// To add items in the Set
127.0.0.1:6379> sadd fruits Apple Mango

// To display all items in Set
127.0.0.1:6379> smembers electronics
```

The following screenshot illustrates these operations in actual environments:


fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help

```
127.0.0.1:6379> sadd fruits Apple Mango
(integer) 2
127.0.0.1:6379> sadd fruits Banana
(integer) 1
127.0.0.1:6379> sadd fruits Mango
(integer) 0
127.0.0.1:6379> smembers fruits
1) "Banana"
2) "Apple"
3) "Mango"
127.0.0.1:6379> spop fruits
"Apple"
127.0.0.1:6379>
```

Basic operations for Redis Sets

These are some of the data types and structures available in Redis to get you started. Let's now move on to the next feature of Redis, that is, the Publish and Subscribe module.

Redis Publish/Subscribe

This module is designed to serve the purpose of a messaging system, where a publisher creates a channel and its subscribers joins those channels. Mostly, in traditional systems, these types of scenarios are handled in a way that the subscriber will query the publisher, asking for any new message after a specific interval. In Redis, whenever a publisher sends a message to its channel, all the subscribers will automatically receive the new message immediately without having to query the publisher and asking for a new message.

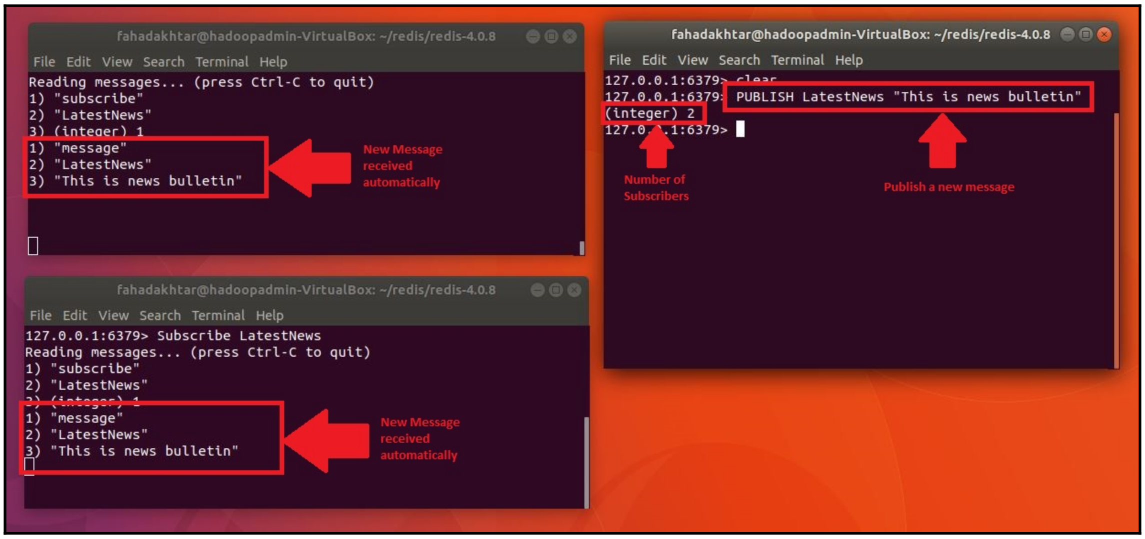
Now let's suppose that the publisher said to its subscriber that it will send all the messages to its channel called `LatestNews`. Subscribe will join this channel by executing the following command:

```
127.0.0.1:6379> SUBSCRIBE LatestNews
```

Publisher will send a message to this channel by executing the following command:

```
127.0.0.1:6379> PUBLISH LatestNews "This is news bulletin"
```

Once the publish command gets executed, all the subscribers of this channel will automatically receive this new message, as illustrated in the following screenshot:


fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help
New Message received automatically
fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help
New Message received automatically
fahadakhtar@hadoopadmin-VirtualBox: ~/redis/redis-4.0.8
File Edit View Search Terminal Help
Number of Subscribers
Publish a new message

```
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "LatestNews"
3) (integer) 1
1) "message"
2) "LatestNews"
3) "This is news bulletin"
```

```
127.0.0.1:6379> Subscribe LatestNews
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "LatestNews"
3) (integer) 1
1) "message"
2) "LatestNews"
3) "This is news bulletin"
```

```
127.0.0.1:6379> clear
127.0.0.1:6379> PUBLISH LatestNews "This is news bulletin"
(integer) 2
127.0.0.1:6379>
```

Redis publish/subscribe in action

If you take a closer look at this screenshot, publisher sends a message to its channel `LatestNews`. Terminal returns `(integer) 2`, which means that the message has been delivered to its two subscribers. At the same time, if you check the subscriber's terminal, they have already received the message, which contains three lines, the first line states that the received type is message, the second line is the channel name, and the third line is the actual message.

Common key operations

The following are some of the key operations that you will be needing while dealing with Redis. In all the following commands, `key` means the datatype such as set, list, and so on against which you want to perform the operation. For example, we have created a list of `fruits`. So `fruit` will become the key.

The following command is required if you want to delete any existing `key`:

```
127.0.0.1:6379> DEL key
```

Execute the following command if you want to check whether any key with the same name already exists or not. If it returns `(integer) 1`, it means true, otherwise it will return `(integer) 0`:

```
127.0.0.1:6379> EXISTS key
```

To set the expiry of any key after a specified interval, execute the following command:

```
127.0.0.1:6379> EXPIRES key seconds
127.0.0.1:6379> PEXPIRE key milliseconds
127.0.0.1:6379> EXPIRESAT key timestamp
```

In order to remove expiration from a key, execute the following command:

```
127.0.0.1:6379> PERSIST key
```

To check the type of `key`, for example, whether the key belongs to List, set, Hash, and so on, execute the following command:

```
127.0.0.1:6379> TYPE key
```

To rename a `key`, execute the following command:

```
127.0.0.1:6379> RENAME key newname
```

These are some of the commands that you will need to perform routine tasks while using Redis.

Summary

In this chapter, we discussed what an API is and its different types. Then we moved on to REST API and dived deep by exploring different types of HTTP request methods involved in it. Then we discussed different types of authentication mechanisms, specifically basic authentication types and JSON web tokens. In basic authentication, we studied how it passes related information in the header of a request. Then we studied different components of JSON web token, such as header, payload, and signature.

We then moved on to the practical part and created a RESTful web service by ourselves. After testing its functionality over the web, we moved on to create its Java client to all the REST web services directly in our program, which went successfully.

Then we started discussing an in memory database called Redis. We saw what it is and why it is popular. We then moved on to its installation. We then set up Redis server and Redis client, which is also called Redis CLI. We then dived into exploring different functionalities of Redis. We then discussed different types of data structures used in Redis such as HashMap, List, Map, and so on. Then we also explored a very popular feature of Redis, which is the Publish and Subscribe module. We also performed a practical of how to publish a message to different subscribers and how subscribers will receive messages. In the end, we discussed different common operations that are helpful in daily activities.

12
Machine Learning

Human beings start learning different things from childhood. The desire to learn increases as we grow older, interact with different people, study different subjects, and so on. It's in the nature of human beings. It gives us the ability to make decisions and predict actions. Computers and machines, however, only do what they are programmed to do, within the range of predefined rules and instructions. They cannot predict or perform an action on their own unless they are programmed to do so. But nowadays, with all the advancement of science and technology, it is becoming a thing of the past that a machine cannot learn on its own. Now, machines have started learning if provided with the correct datasets. It is similar to the childhood of a human being, but with all of the new research and advancement, it all will grow very quickly.

In this chapter, we will discuss the following topics:

- Machine learning
- Types of machine learning algorithms
- Supervised machine learning model
    - Classification model
    - Regression model
- Unsupervised machine learning model
    - Clustering, k-means
    - Neural network
    - Deep neural network
- Decision tree classifier

Let's start our discussion with a question: what actually is machine learning?

Machine learning

Machine learning is a field of computer science and a subsection of artificial intelligence. Machine learning gives a computer the ability to learn things by itself. It is based on many different algorithms that learn things by experience and interaction instead of predefined rules.

Machine learning (**ML**) is very closely linked with Probability and Statistics, a mathematics field. ML uses its many algorithms and formulas for prediction and the analysis of data. Machine learning is all about data and the algorithms used to transform it, learn, and get trained on it.

Some of the naming conventions in machine learning are actually the same in nature. For example, the independent variable may also be referred to as a predictor variable. Another example is target variable; you may also see it referred to as a
dependent variable or response variable.

Now let's move the discussion to our next topic, which is about the types of algorithms used in machine learning.

Types of algorithms

Machine learning is all about adaptation and learning. This learning is purely based on the algorithm that we apply in our model to get the desired result. These algorithms can broadly be divided into two types:

- Parametric algorithms
- Non-parametric algorithms

Let's now start our discussion with the first type, parametric algorithms.

Parametric algorithms

In this type of algorithm, most of the things are assumed beforehand. On one hand, it simplifies the learning process, whereas on the other hand, it limits the number of items that can be learned. Let's take an example of image recognition. We have already assumed that the input image will be of an animal with four legs and a tail. By this assumption, it limits the computer to learning and predicting other types of animals and infecting any other type of object. Examples of this types of algorithm include perceptron, linear regression, linear discriminant analysis, and so on.

The main advantage we get from these types of algorithm is simplicity. It also speeds up the learning process and it requires less data for the learning purpose. The main disadvantage of it is that it limits the model's ability to learn new things.

Non-parametric algorithms

Unlike parametric algorithms, in this type of algorithm, we don't make any weak assumptions. It may be because of the lack of knowledge that we may have about the data or it may be because we want our model to be flexible and let it learn and get trained for new things. Whatever the reason we choose this type of algorithm, it helps the model to adapt to it's environment and changes by itself. Examples of these types of algorithms include k-nearest neighbors, support vector machines, decision trees, and so on.

The main advantage of using these algorithms is that they are capable of learning new things and not limited by our assumptions. The other advantage is that once it gets trained, they will perform much better. The main disadvantage of using these algorithms is that they require a lot more data compared to parametric algorithms. As there is much more data, they will require more time to get trained.

In machine learning, we train our model to predict different outputs. The type of training that we perform in our models can be classified into two categories; supervised learning and unsupervised learning. Now let's explore what these are by starting with the first category, supervised learning.

Supervised learning

In this type of learning, computers learn from a predefined data set with data labels and features. Its aim is to predict an output based on the given input variables using the defined data point and label from the learned dataset. The most important thing you need in supervised learning is data with labels. By providing data labels, we teach and train our model for accuracy. The more accurate your data is, the closer your prediction will be. Some of the ways of getting data include from a historical source, or by doing experiments.

Now let's take an example of supervised learning. In your mailbox, you have a folder for junk emails; ever wondered how it automatically identifies emails as spam? It is actually based on the trained model, which looks for certain things before marking it as junk, including the source from where the email was generated, the intended audience (whether it is directly targeting the recipient), whether the email body contains marketing or spam keywords, and so on. Smart spam filters also keep learning based on user interaction; for example, if per the filter an email is not spam but the end user marks it as spam, this means that something has been missed out by the filter. Therefore, the filter retrains itself after some interval, considering all of these scenarios.

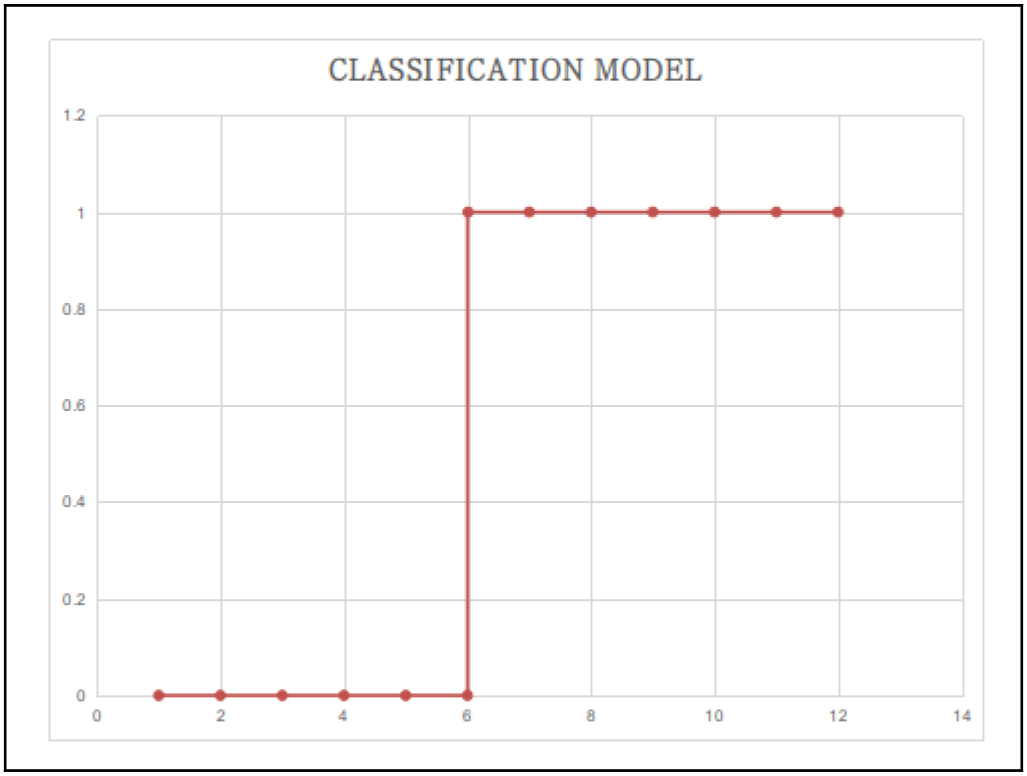
There are mainly two types of learning models in supervised learning; the first one is the classification model and the second one is the regression model.

The classification model

The classification model mainly gets trained to predict fewer output data labels. For example, answers involving prediction of yes or no, spam or not spam, whether a picture contains a dog or a cat, and so on. The resulting output must be few and its values must be discrete. There are two type of classification models; binary classification and multi-class classification.

Binary classification

As the name states, this model only predicts either of the two values. It may be yes or no, *0* or *1*. Let's take an example of a blood test. After conducting a study, it was found that if the value of sugar is less than *6* mmol while fasting, then the person doesn't have diabetes. Let's build our classification model based on the collected data, as shown in the following diagram:



Classification model to diagnose whether a person is diabetic

Now, once we pass a blood sugar test value to our classification model, it will easily predict whether a person is diabetic or not.

Multi-class classification

This model is used to classify from a category of output based on the provided inputs. For example, using this model, we can predict whether an image contains an animal or not. It may be used to identify whether it's a dog or a cat.

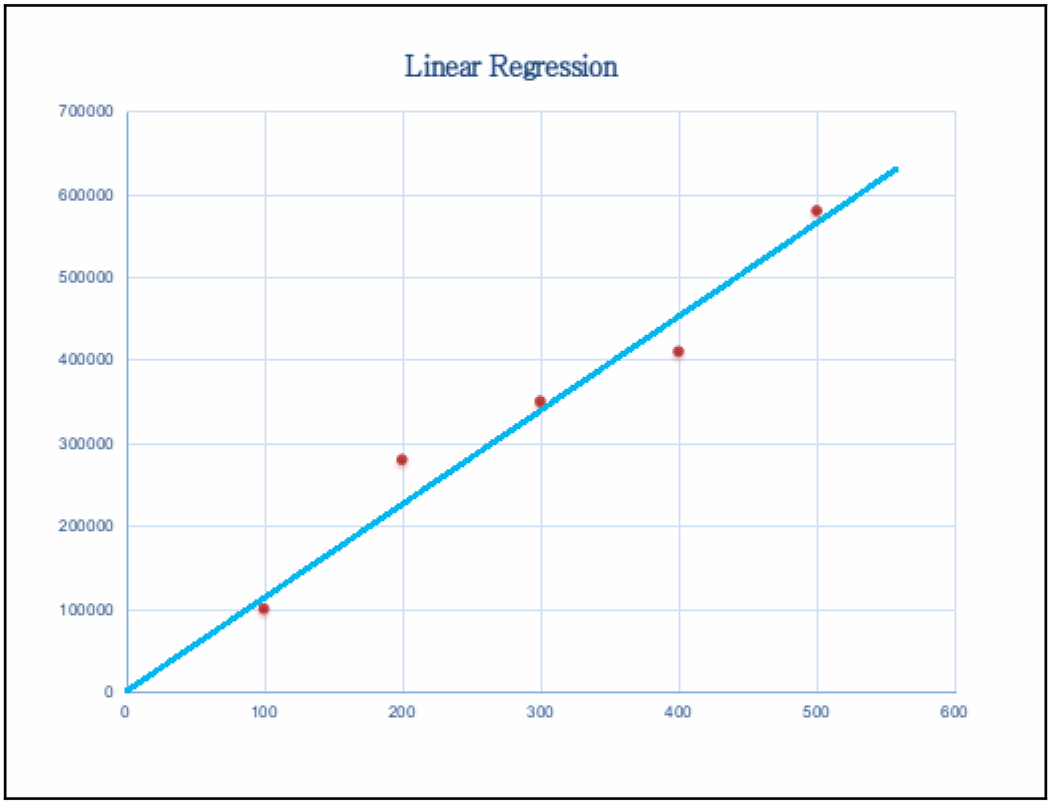
Let's move on to a more complex topic where we have to predict different range values instead of a simple yes/no or a category.

The regression model

This type of model has many different output labels based on the varying input variables. This model is mainly trained to predict continuous non-discrete values. There are different types of regression models, but in machine learning you will be dealing mainly with two types, linear regression and polynomial regression.

Linear regression

Let's suppose we have collected data related to the size of a house and how much it costs. We then plotted the values in the following chart. Each dot represents the size of the house and its value. Now, to predict the price of a house of any given size, we will draw a straight line similar to the one in the following figure. The characteristic of this line is that it is drawn in such a way that it is very close to most of the plotted values. Some values may be above or below the line, but it touches most of the values. So, the closer this line is to the values, the more correct our predictions will become:

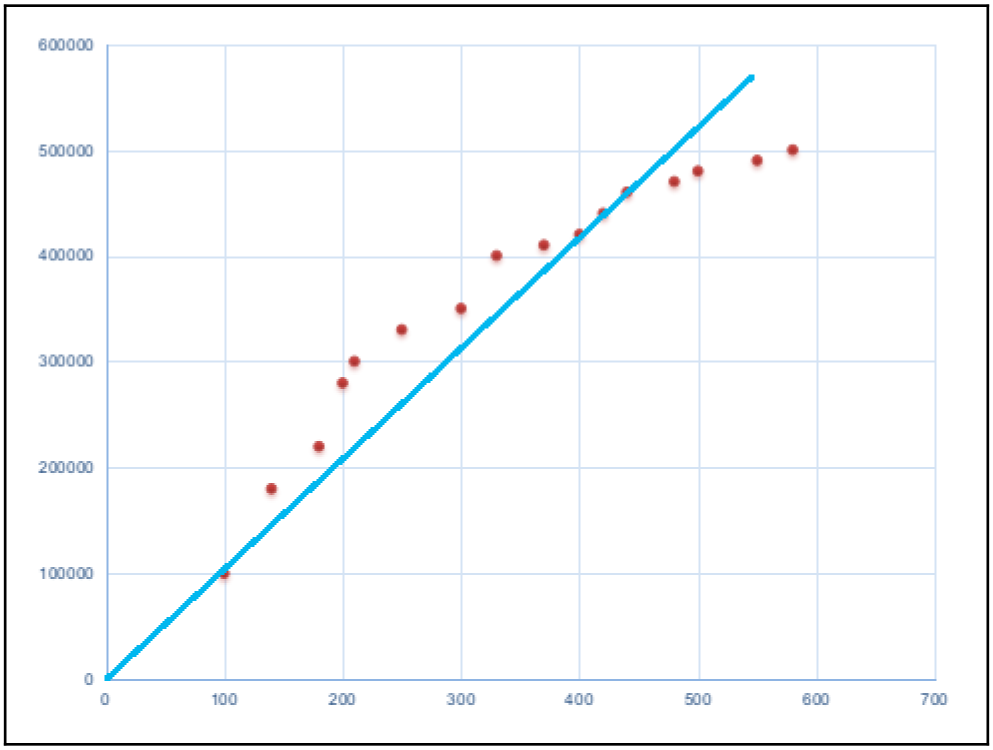


Linear regression of price versus the size of a house

Now, by having this linear regression model, we can easily predict the price of a house by using its size. Let's suppose, if you have to say, what would be the price of a 150 square yard house? By using this model, it will predict that the price will be somewhere around $150,000.

Polynomial regression

Polynomial regression is also referred to as **curvilinear regression**. Let's start by taking the same example of a house and its price. In the last section, we only considered using the size of a house to predict its price. But now, let's suppose we have other features to consider in our observation. For example, the number of bedrooms in a house, whether it has a swimming pool or not, and what the actual constructed area of a house's land is. Now, with all these features in mind, plot these values in a graph. Now, try to place a line close to the plotted values. You will end up with a chart similar to the one in the following diagram:



Linear regression application

Now, observe this chart; initially, the values increase drastically as the size increases. It may be because of different features, but after a certain size, the prices are not jumping too high. So if we apply linear regression in this scenario, we may end up predicting the wrong price of the house.

For these types of scenario where linear regression is not providing the correct output, we use polynomial regression. There are mainly two types of polynomial regression, quadratic and cubical. Quadratic regression is the sum of features multiplied by the input variable. Mathematically, it is represented using:

$$y = a_0 + a_1 x + a_2 x + \ldots\ldots\ldots\ldots + a_n x$$

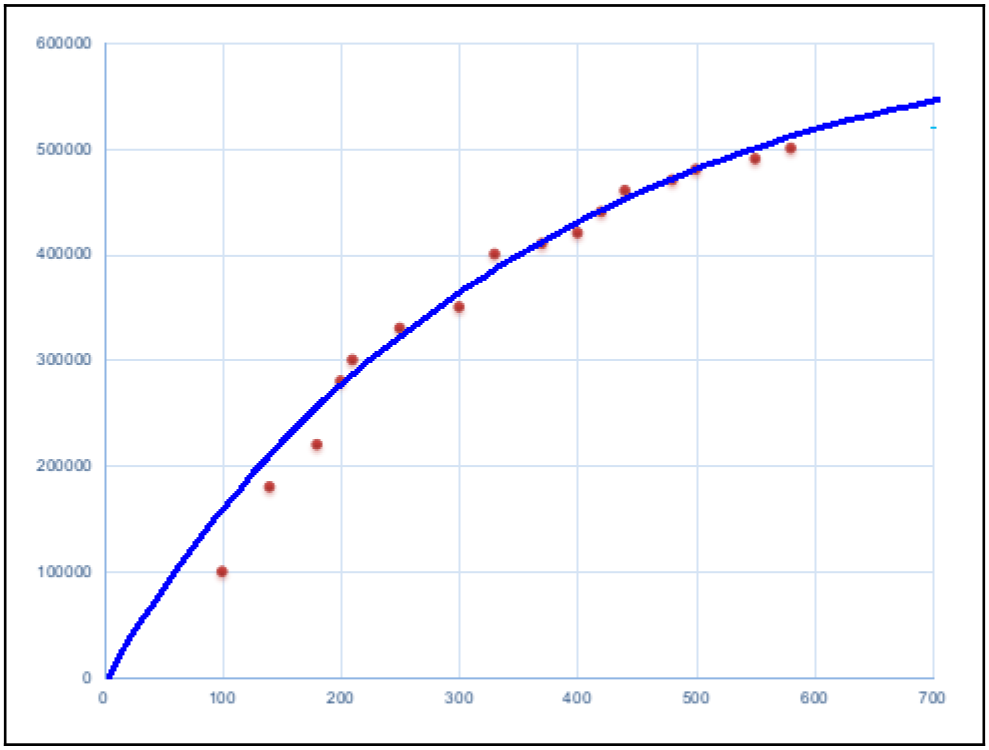
Quadratic regression formula

Cubical regression is also the sum of features multiplied by the input variable, but with each increasing feature, the power of the input variable also increases. Mathematically, it can be represented as follows:

$$y = a_0 + a_1 x + a_2 x2 + a_3 x^3 + \ldots\ldots\ldots + a_n x^n$$

Cubical regression formula

In the current given scenario, if we apply a quadratic formula, the prices will start descending after a certain size of house. This will never happen in a real life scenario, where the size of the house is bigger and the price is lower. We will apply a cubical formula in this case. Let's re-plot the values and apply polynomial regression using, cubical formula. We will get a chart that looks like the following:



Polynomial regression

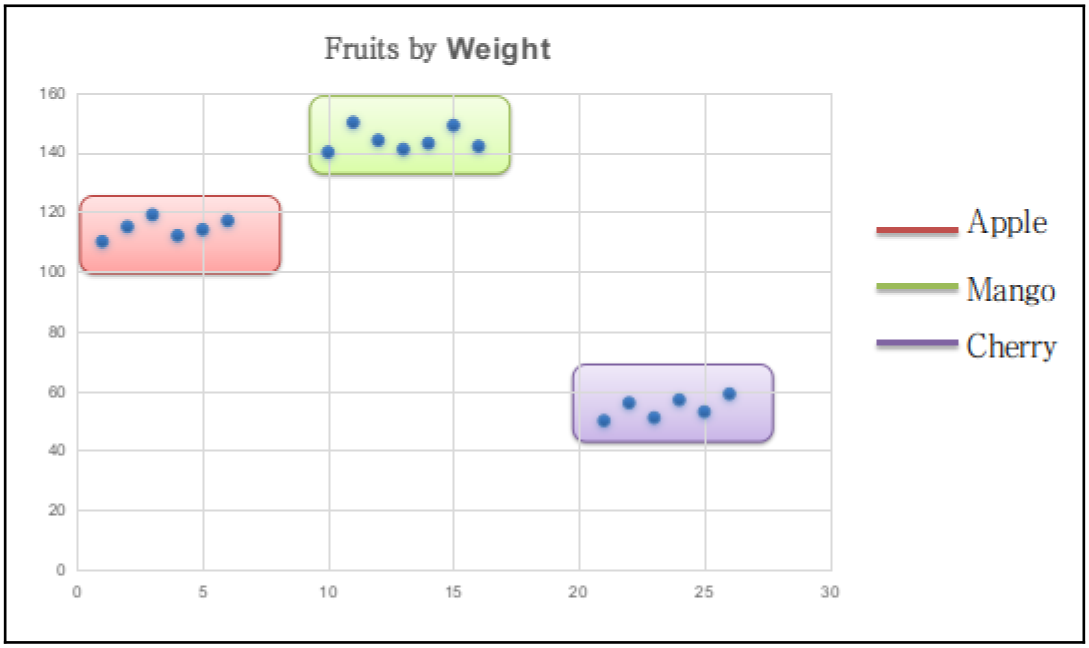
In this chart, the line is closer to the plotted values, thus it will better predict the output values of a given house.

Now let's move on to the next learning type in machine learning, which is unsupervised learning.

Unsupervised learning

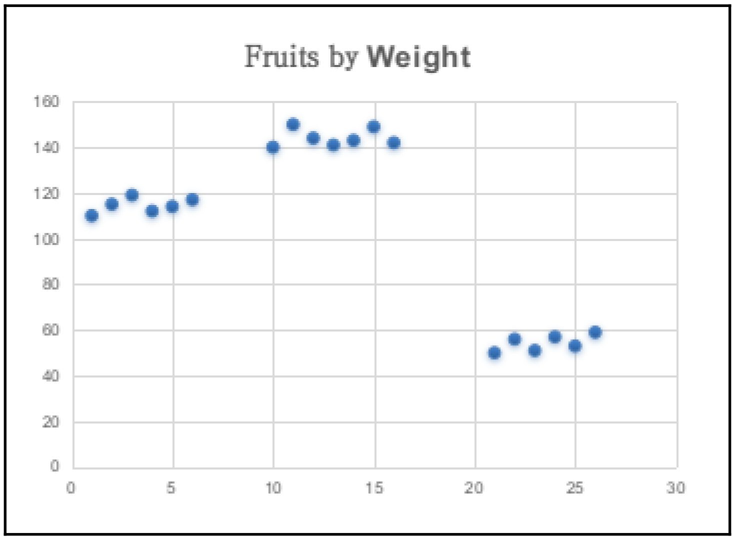
Unsupervised learning is a fairly complicated learning model compared to supervised learning, in which the data that we are providing to the model to learn from is not labeled. It means that by reading the data, it cannot tell whether it belongs to class A or B, or if it's related to male or female, and so on. In these types of scenario, where the data is not labeled, we apply different algorithms so that a machine can at least distinguish between different sets/classes/categories within the provided data.

To understand it more clearly, let's take an example dataset to distinguish between different fruits based on their weights. By looking at the following chart, one can easily tell that if the weight is between 140 to 150 grams, it's a mango. Similarly, if the weight is between 110 to 120 grams, it's an apple, and if the weight is between 50 to 60 grams, it's a cherry:



Fruits dataset by weight

But let's suppose that there are no labels in the provided data. How can one tell which one is which, as shown in the following figure:



Data with no labels

Here, unsupervised learning comes into the picture, where it applies different algorithms in order to identify different formations and patterns in the data. The following are the most commonly used algorithms in unsupervised machine learning:

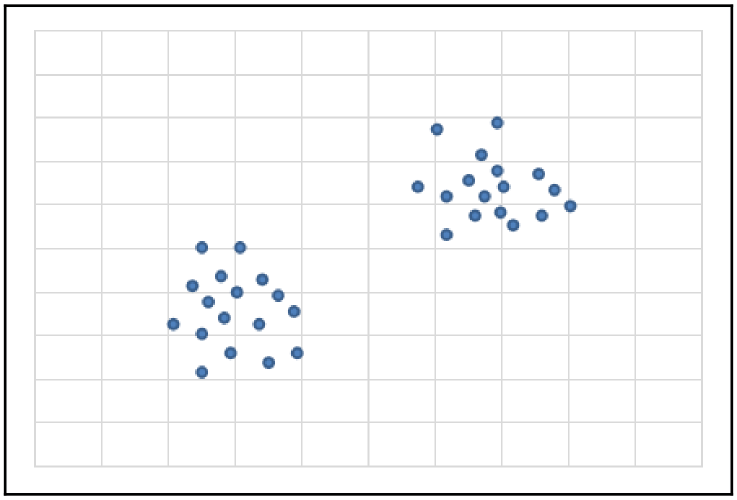
- Clustering, k-means algorithm
- Neural networks

Let's start by learning more about clustering using the k-means algorithm.

Clustering, k-means

We have already established the understanding that in unsupervised learning, data is not labeled. A machine cannot tell by itself that it's a fruit, flower, human, or anything, so it makes its own groups and classes to distinguish data from one class to another. A clustering algorithm, as the name states, helps to identify clusters within the provided data. It actually divides the data into logical groups based on the provided parameters. After identifying the required number of clusters, it will predict the future occurrences of these groups in the next available data.

Now let's understand this more clearly using the following example. Assume we have the following plotted data:

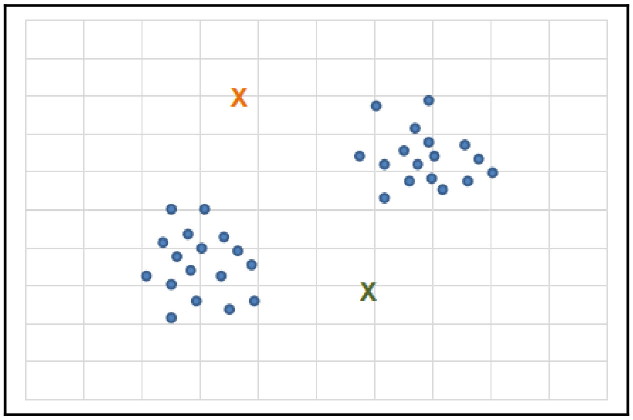


Data to apply the cluster k-means algorithm

Now, when we see this image with the naked eye, it can easily identify the clusters within this image. But how will a computer recognize it? For this purpose, we will apply the k-means algorithm. It states the following:

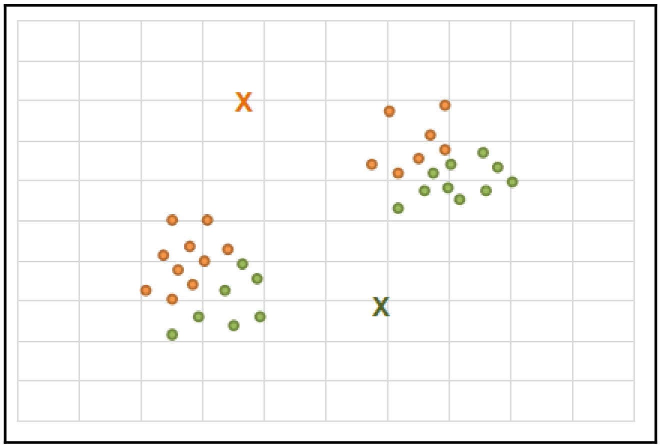
1. Randomly plot the cluster points in the same manner as the required clusters to be identified. In our case, we want to identify two clusters.
2. Calculate the distance from these cluster points with the available data points.
3. Associate the closest data points with the respective cluster points.
4. Calculate the mean of the associated data point for each cluster point.
5. Move the cluster points to the mean value calculated in the previous step.
6. Repeat steps 2 to 6 until the cluster points stop changing their places.

Let's apply this algorithm in a graphical way for ease of understanding. Follow point number 1 in the preceding list and randomly plot two cluster points. The first one is **X** in an orange color, and the other one is **X** in a green color, as shown in the following diagram:



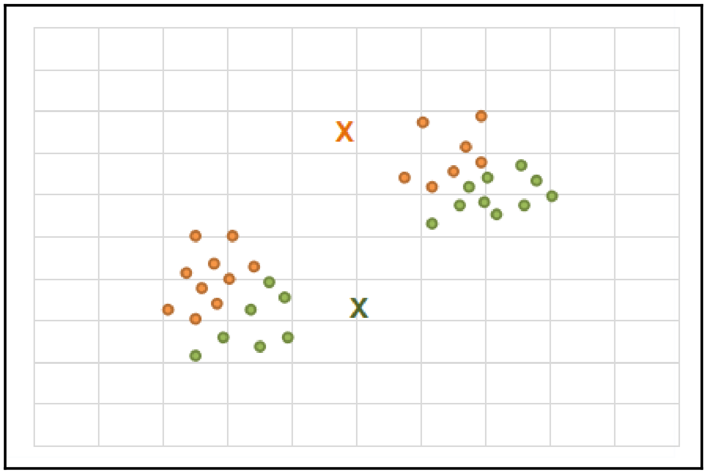
Randomly plotting the cluster points

The second step is to calculate the data point distance from the cluster points and associate it with the relative cluster point. To illustrate this in a graphical way, the color of the data points closer to the orange **X** point has been changed to orange. Similarly, the data points closer to the green **X** cluster point have been changed to green, as shown in the following diagram:



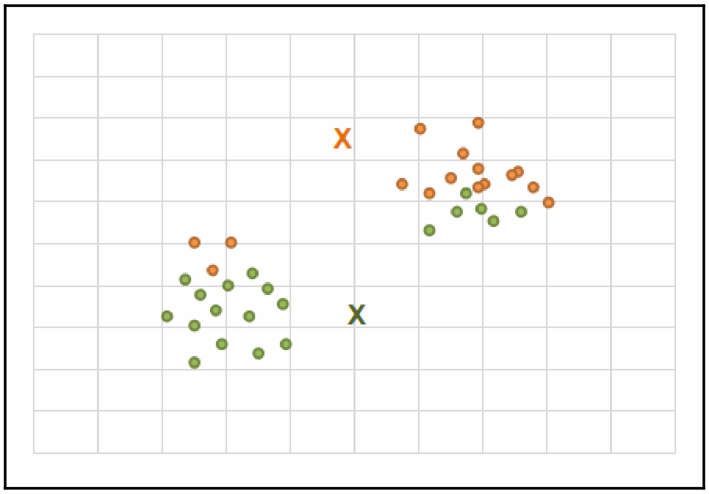
Associated data point closer to the respective cluster point

The next step is to calculate the mean of the related data point associated with the cluster point and then shift the cluster point to the calculated value, as shown in the following diagram:



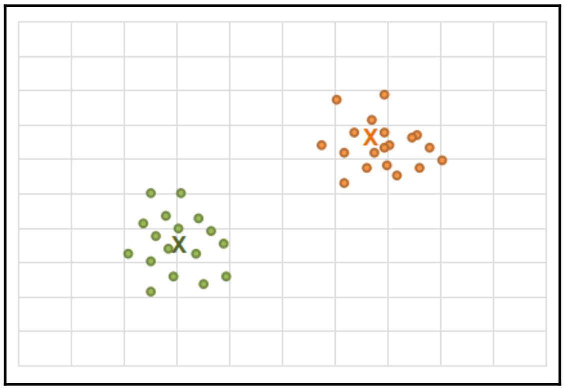
Shifting cluster point based on the calculated mean

The next step is to recalculate the distance of the data point with respect to the cluster points, as shown in the following diagram:



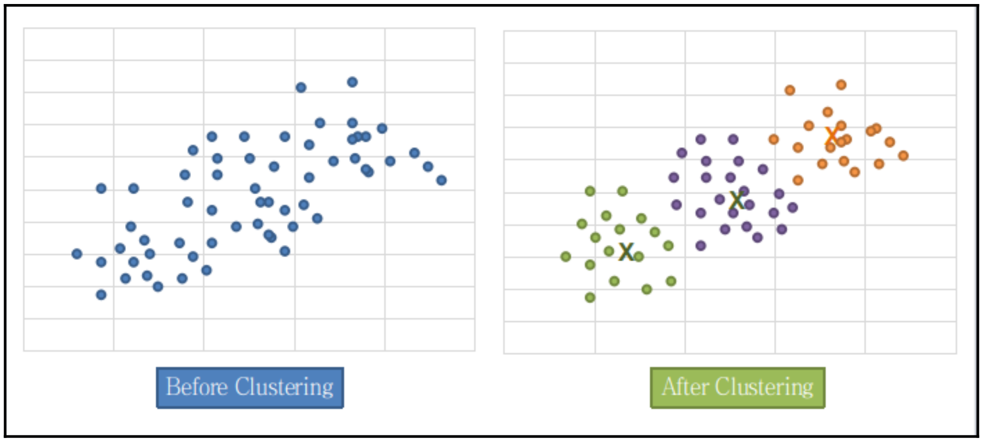
Re-association of the data points with respective cluster points

Repeats these steps until the cluster points are no longer moving and clearly declare two clusters, as shown in the following diagram:



Final result of the k-means cluster algorithm

This example used the k-means algorithm to help the machine identify different clusters of data in an unsupervised environment. The important thing to note is that the data doesn't have to be clearly separated to form a cluster. If you have scattered data with no clear clustering, you can still apply the k-means algorithm to divide it into clusters. The following is just an illustration of such a scenario:



Cluster application on scattered data

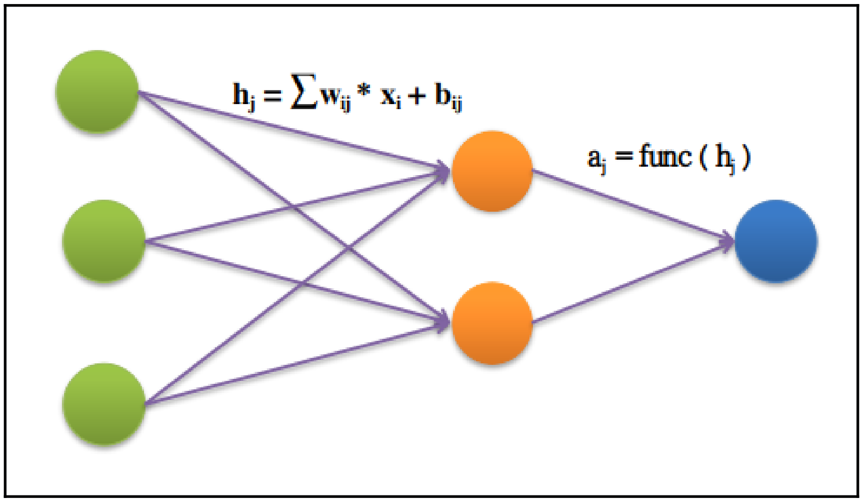
As you can see in this image, the data doesn't seem to be divided into clusters, but when we apply k-means, it gets three clusters as an output.

Let's move on to our next topic on the journey towards understanding unsupervised learning, which is neural networks.

Neural networks

This learning mechanism is inspired by how a human brain works. A human brain is made up of billions of nerve cells called **neurons**. An estimation is that there is a range from 50 to 500 billion neurons in human brain. They are very well interconnected to each other with many to many relationships, which means that one neuron is not just connected to one neuron but with multiple neurons.

In a neural network, the neurons are called **units**. There are some input units. These input units interact with the outside world and receive data from it. They will then send their data to an inner layer, also known as the hidden layer. The important thing to note here is that each input unit sends its data to all the units in the hidden layer. The number of hidden layers varies from one to n-numbers, depending on the type and purpose of the neural network. These hidden layers are the actual brain of this neural network. The units of these hidden layers will then process and learn things and produce their data to the output layer. An illustration of a typical neural network is shown in the following diagram:



A neural network

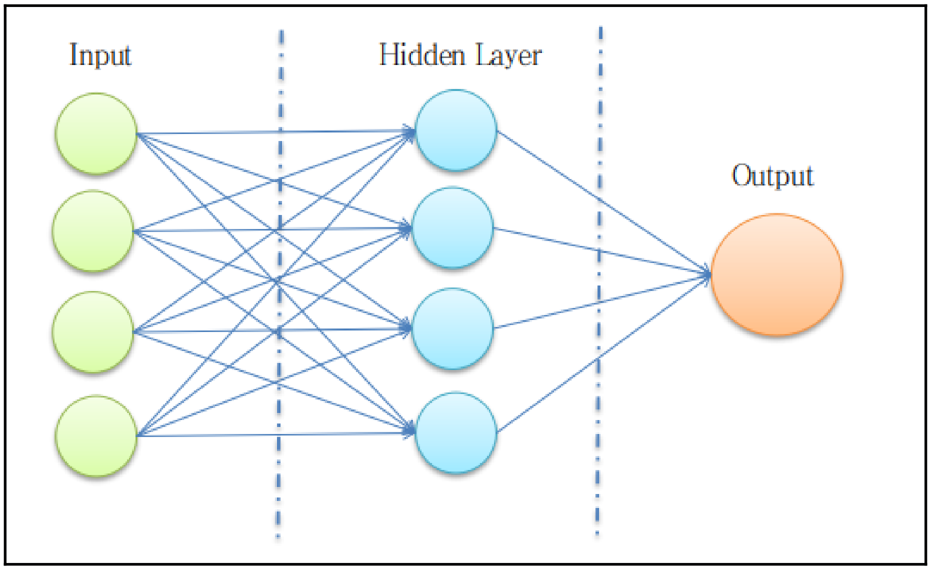
The following are the main types of neural networks:

- Feedforward neural networks
- Recurrent neural networks
- Symmetrically connected neural networks

Let's explore them one by one in the following subsequent sections.

Feedforward neural network

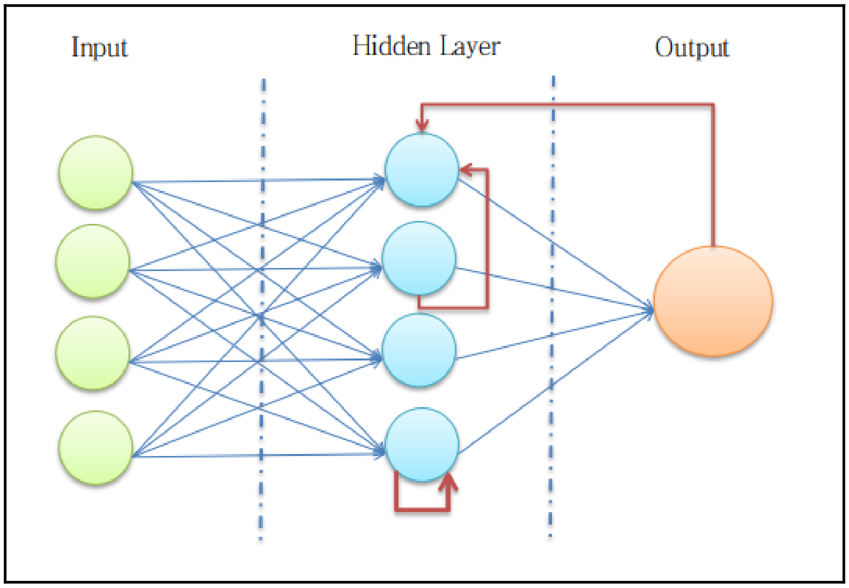
In this type of network, the information flows in one direction. Typically, it receives an input from the outside world and passes it on to the hidden layer for whatever purpose they exist for. Afterwards, the hidden layer passes the resulting value to the output unit. The following diagram shows a typical feedforward neural network architecture with one hidden layer. The hidden layer may vary depending on the design and requirements of the neural network:



Feedforward neural network architecture

Recurrent neural network

In this type of network, input units pass on data to the hidden layer. Then, the hidden layer may use their outputs as an input to the same layer. This means that one unit of the hidden layer, instead of sending the data to the output layer, may send it to other units in the same hidden layer, or back to itself as input. It is also possible that the hidden layer sends the data to the output unit and the output unit will then send the data back to the hidden layer. Sometimes, it leads you back to the unit where you start giving inputs. An illustration of these examples is shown in the following diagram:



Recurrent neural network architecture

These types of networks are very good at remembering things for a very long time and are very similar to our biological brains. On the other hand, their implementation is very hard and complex.

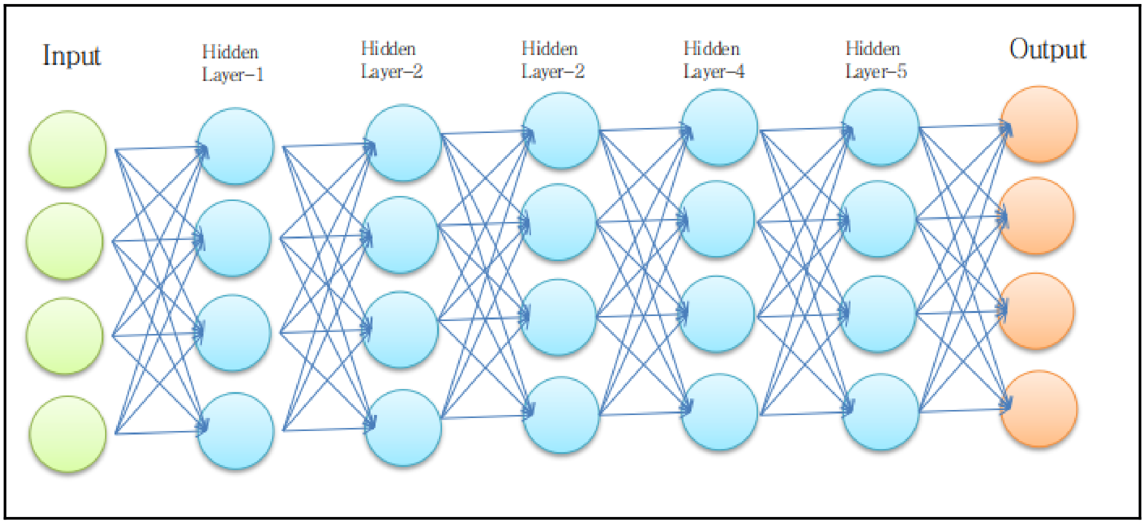
Symmetrically connected neural network

A symmetrically connected neural network is very much similar to a recurrent neural network. The only difference is that it has symmetrical weights in both directions of the unit, hence the name symmetrically connected network.

If the symmetrically connected network has hidden units, it's called a **Boltzmann machine** and if it doesn't have hidden units, then it's called a **Hopfield network**.

Deep neural networks

A neural network can consist of n-hidden layers. Typically, if a neural network model consists of more than three layers, it is considered to be a deep learning model. The following diagram illustrates the architecture of such a network:



Deep neural network architecture

As of now, the deepest neural network is designed by Google, and has 22 hidden layers. Imagine the amount of complexity and computing it requires to handle data.

Let's move on to our next topic, the decision tree classifier.

Decision tree classifiers

This algorithm is used to predict an output based on the available labeled data having varying parameters. To deal with the problem, it builds a tree-like structure that has nodes to distinguish different characteristics within data.

Let's take an example of a boy named Smith. We have collected data related to Smith's outdoor activities that can tell us which weather condition is most likely to provoke Smith into going out to play with his friends. The following is the data collected over a period of 10 days:

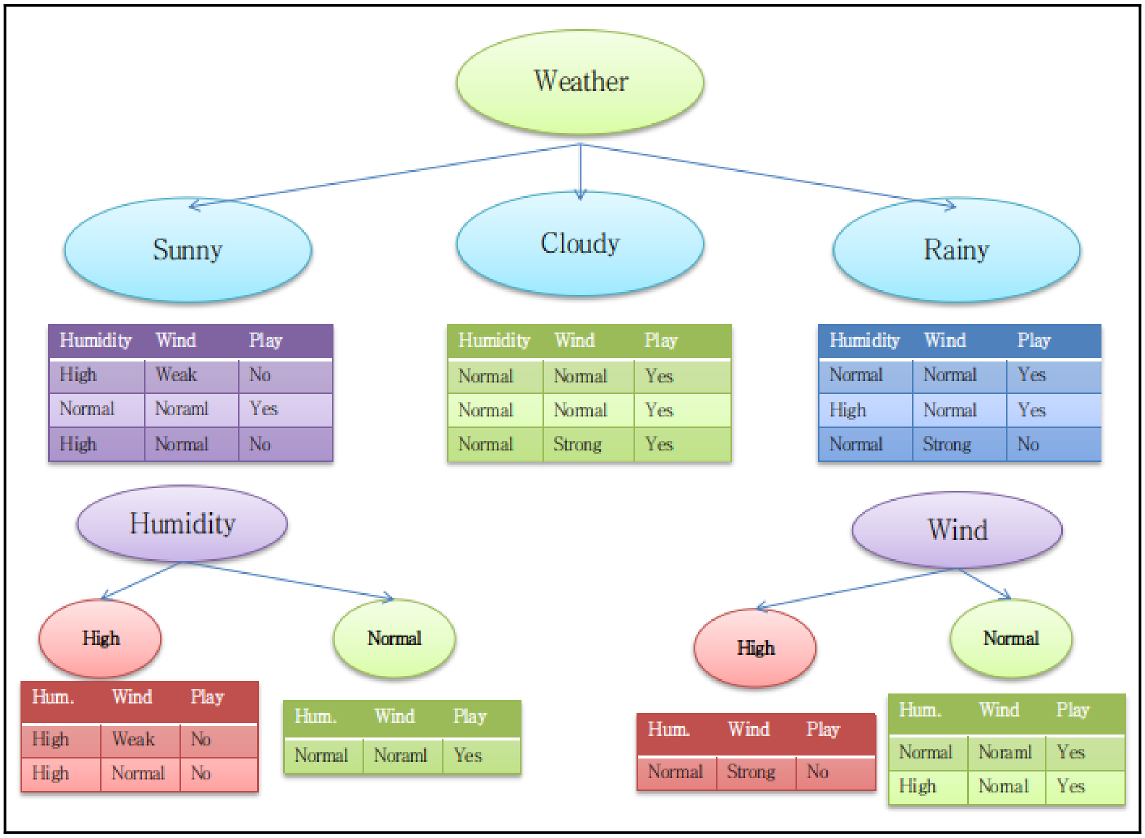
Weather	**Wind**	**Humidity**	**Played**
Cloudy	Normal	Normal	Yes
Sunny	Weak	High	No
Sunny	Normal	Normal	Yes
Cloudy	Normal	Normal	Yes
Rainy	Normal	Normal	Yes
Sunny	Normal	High	No
Rainy	Normal	High	Yes
Rainy	Strong	Normal	No
Cloudy	Strong	Normal	Yes

Decision trees are built on a series of questions about the data. For example, if we look into the current data, we might have the following possible questions:

- What is the weather like?
- How is the wind blowing?
- How is the humidity?

In the decision tree, the most important thing is what question to ask and when. A decision tree's structure starts with a question, but what question? Shall we start by asking about humidity or weather? To solve this issue, we will calculate the impurity of the actual data. We than have to iterate through each value using all of the possible question candidates. We then partition them based on the uncertainty of data and their impurities. We care more about a large dataset with low impurity that a small set with high impurity. Then, we subtract the resulting impurity with the actual impurity. Finally, whichever candidate has the lowest impurity is the best question to ask.

We will now build our decision tree, as shown in the following diagram:



Humidity	Wind	Play
High	Weak	No
Normal	Noraml	Yes
High	Normal	No

Humidity	Wind	Play
Normal	Normal	Yes
Normal	Normal	Yes
Normal	Strong	Yes

Humidity	Wind	Play
Normal	Normal	Yes
High	Normal	Yes
Normal	Strong	No

Hum.	Wind	Play
High	Weak	No
High	Normal	No

Hum.	Wind	Play
Normal	Noraml	Yes

Hum.	Wind	Play
Normal	Strong	No

Hum.	Wind	Play
Normal	Noraml	Yes
High	Nomal	Yes

Decision tree architecture based on the given data

Now that we have trained our model based on the provided data, we will now ask a question to our model and it will predict Smith's next decision - whether he will go out to play or not. Let's assume the weather is sunny, the humidity is high, and the wind is normal. Our decision tree will check for the first question that it was built upon; what is the whether like? Based on the provided data, the answer is Sunny. The next question the decision tree will ask is about the humidity; is it High or normal? The answer is High. Now, the decision tree will not ask any further questions, and will predict that Smith is most likely not to go out to play.

Summary

In this chapter, we learned about machine learning. We then learned about the different types of algorithm it uses to get the desired information. We then had a discussion about two types of model, the supervised model and the unsupervised model, which are used to train machines to predict their input.

We then had a discussion about what supervised learning is. We then discussed the two commonly used algorithms in supervised learning. The first algorithm we discussed is called the classification model; here, we discussed binary classification and multi-class classification. The second algorithm we discussed in supervised learning is called regression; here, we discussed linear regression and polynomial regression.

We then had a discussion about what unsupervised learning is. We then discussed the two commonly used techniques in unsupervised learning. The first technique is called clustering, which uses the K-Mean algorithm to identify patterns in data. The second technique is called a neural network; herein we studied its three types, the feedforward neural network, the recurrent neural network, and symmetrically connected neural networks. We then had a discussion about deep neural networks.

In the next chapter, we will learn about **artificial intelligence** (**AI**) and the most important topic in AI: **convolutional neural networks** (**CNN**).

13
Artificial Intelligence

Human beings are the smartest species. No other species reaches to the level of intelligence that we have. It is because of this intelligence that we feel the need to create things that will help us in our daily life, ease our work, and help us achieve more in less time. To pursue this, we as human beings invent so many things from time to time for the sake of advancement, to improve our lifestyle, and to ease our day-to-day tasks. On this road of inventions, the most important thing that we came up with was electricity. It opened up the whole new world of possibilities that we can achieve.

We have designed a lot of machines that run on electricity in order to perform our required tasks. Whatever we programmed them to do, they do, but still we need more from it. Curiosity is the integral part of human nature. It leads us to think about machines in a different way. So, we started the thought process, what if it can decide on its own instead of doing the programmed task with pre-defined rules? What if it can function similar to a human being? What if we put thought processes into it so that it can learn on its own from the environment around it?

In this chapter, we will cover the following topics:

- Artificial intelligence
- Convolutional neural networks
- Deep learning using TensorFlow
- Object detection using YOLO

Let's start our discussion about artificial intelligence.

Artificial intelligence

Artificial intelligence is intelligence that doesn't come by default from nature. It is created artificially to make machines intelligent. With the help of continuous research and rapid advancement, we are now able to make machines smarter, so that they can adopt to their environment and make decisions to fulfill the purpose they are designed for.

If we go into history, some time in the 1940s, when the first computer was invented, it was considered a very big achievement in the history of invention. At that time, the size of the computer was bigger than a room. The memory was in bytes and Kilobytes and the energy it consumed could light up a whole town. Then, in the 1950's, a group of scientists presented a theory that they could make machines so smart that they could think, learn, and act like humans. They got millions of dollars in grants to make it happen. But at that time, the limitation of computer hardware become the hurdle that they were not able to solve.

Nowadays, computer hardware is not an issue anymore. Even in our home, we have a laptop that has a memory of gigabytes, hard disk capacity in terabytes, and quad core processors in GHz, and we can easily carry them in our bags. Nowadays, even a high end mobile phone has computational power and memory similar to a laptop. In short, we can achieve what we were not able to do in the past because of hardware limitations.

The field of artificial intelligence is an active field of research. It is not just about robots walking among humans, acting like humans. It is everywhere, from the Google search engine to YouTube suggested videos, from self-driving cars to autonomous robots, from speech recognition to face recognition. In fact, nowadays, more and more companies are adopting AI to boost their business and to automate things as much as possible.

From almost the last decade, the active field of research is the human brain and how it works. There are continuous studies and research projects going on that aim to simulate a human brain in a machine, so that the machine can also become human-ish.

An example of Artificial Intelligence is Google AlphaGO. It's an AI program that was created to play the ancient Chinese game **GO**. It's a strategy board game. It received training by playing with different human players and learned from the strategies they applied and the mistakes both made. It then learned different strategies by even playing with itself. The maker of AlphaGO then challenged an expert player of this game in Korea to a 5 match series. This player's track record was amazing; he was almost undefeatable. AlphaGO was able to defeat him 4-1. After the match, many experts in **GO** were shocked and surprised, as AlphaGO applied many strategies to the game that humans had never thought of before.

One of the core concepts of artificial intelligence that has lead to this much advancement is the concept of **convolution neural networks** (**CNN**). Let's find out what they are and why they are so important.

Convolutional neural networks

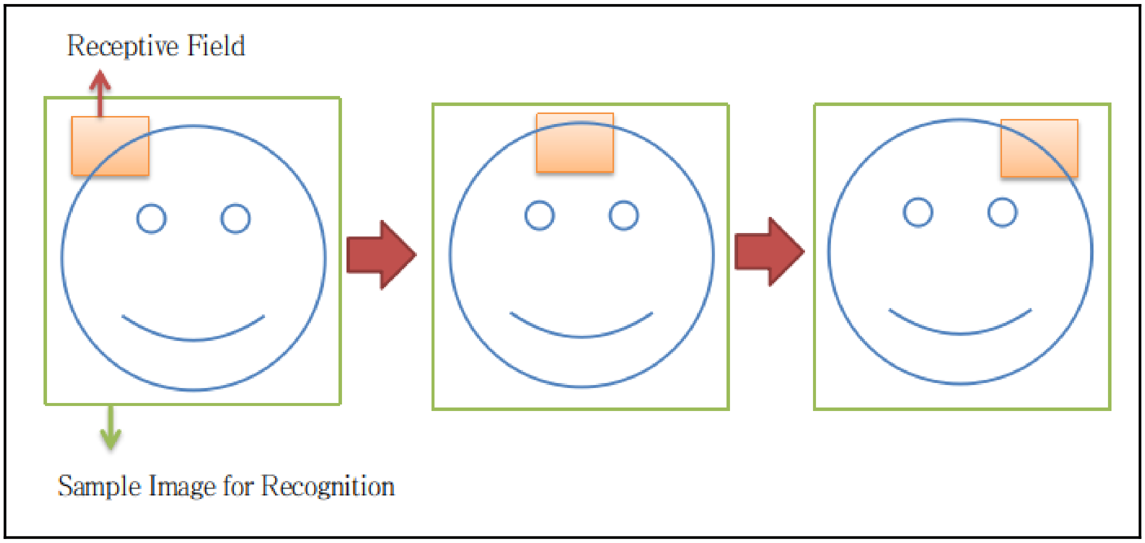
This model is derived from the research of two researchers, D. H. Hubel and T. N. Wiesel. In their paper, they explain how any mammal interacts with the objects around them, and in turn, what happens to the neurons in their brain. Using the same hypotheses and theory, computer scientists try to build the same biological neural structure and mechanism in order to detect and recognize images. This neural network is called a **convolutional neural network (CNN)**.

When a human sees an object, for example, a car, they can tell that it's a car, no matter what, whether they are looking at the front of it, the side of it, the back of it, or the top of it. They are still be able to recognize that it's a car. The research paper states that there are mainly three characteristics or features of a mammalian visual cortex that help identify an object. These three characteristics are:

- Local connections, which means how a cluster of neurons is connected to another cluster or set of neurons; together they represent a feature.
- Layering, meaning that there is a hierarchy of features to be learned by dividing an object into sets of different layers.
- Spatial invariance, which means that no matter from which angle you are looking at an object, it still means the same thing. For example, if you see a banana in an upside down position, you will still recognize it as a banana.

These are the three features of a mammalian visual cortex that inspired the creation of a convolutional neural network, so that a model similar to the biological one can be programmed to act as one.

Let's take an example image of a smiley face. When we pass it on to a convolutional network, it takes a piece of an image called the Receptive Field. We take the dot product of this receptive field matrix and moves the receptive field to the next selection, as shown in the following diagram:

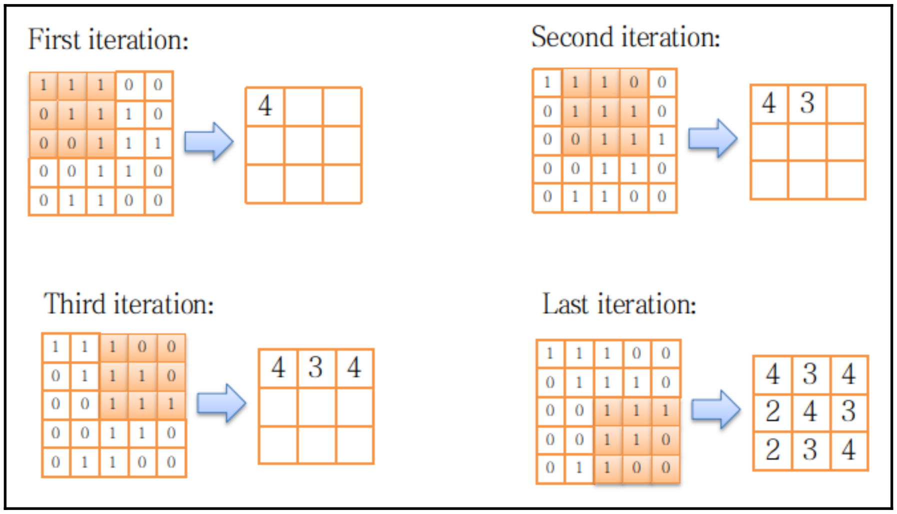


Movement of receptive field in an image

We perform a series of operations in each layer. But first, let us broadly categorize convolutional networks into two categories; the first category is called feature learning and the second category is called classification. The feature learning category almost contains 80% of the convolutional network, whereas the classification category contains the remaining 20% of the network.

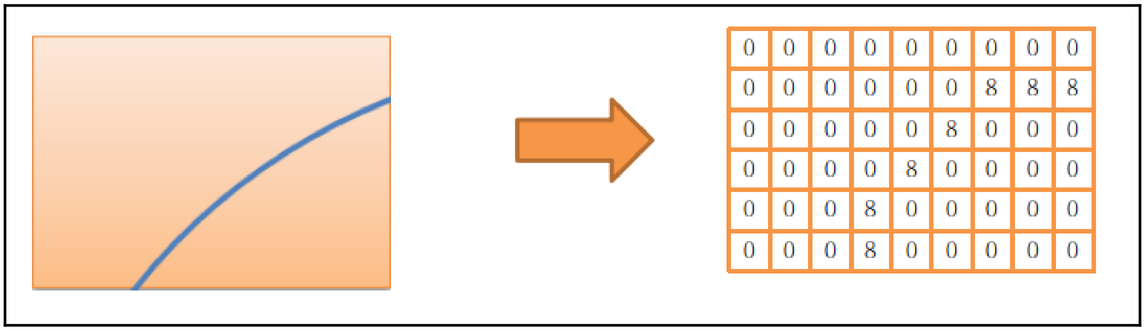
In feature learning, CNN performs an operation over and over, which is also called **convolutional blocks**. These blocks perform three operations: convolution, **ReLU** (rectified linear unit), or any kind of activation function and pooling.

A convolution operation performs a matrix dot product with the weight matrix into the input layer and passes on the output to the next layer. It keeps taking the dot product of these matrices to generate a new matrix in order to detect a feature. The illustration of this process is shown in the following diagram:



Convolution iteration for feature detection

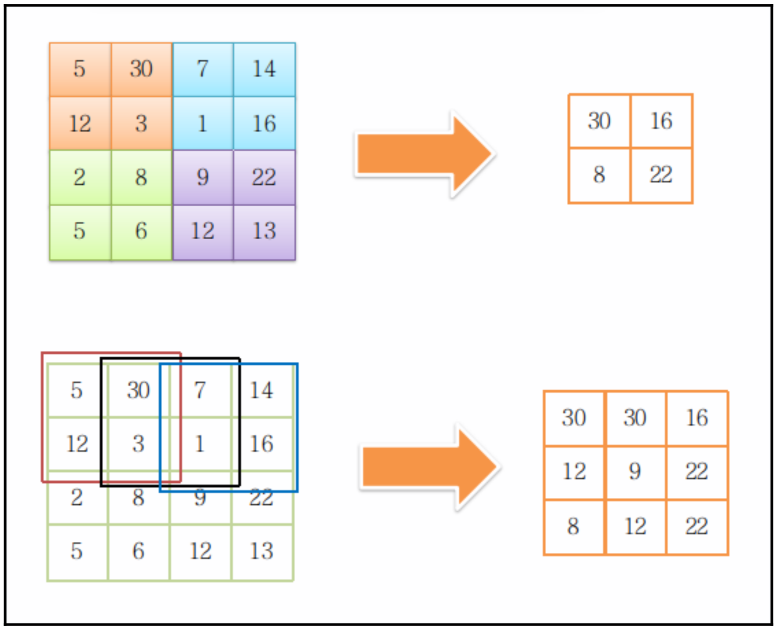
Now, back to our example of a smiley face. The convolution layer forms a matrix of the receptive field, as shown in the following figure. It contains lots of zeroes and numbers, but only on the places where it found a drawing line. It will then compare it with the features of different objects that our trained model has already learned from the label data:



Feature representation in a matrix form

For the comparison part, it will multiply the input matrix, which is shown in the diagram, and the matrix from the already learned features. If the result comes up with some high number, it means that the current receptive field feature has been found within the already learned features. If the resultant number is zero, then the current feature is not available within the learned dataset.

Once the convolution step is completed, the resulted matrix will be passed on to the next layer, which is pooling. The most common type of pooling that we do is max pooling. What this does is pick the highest or maximum value from within a defined window within a matrix. The following diagram illustrates the max pooling mechanism:



Max pooling mechanism

Commonly, there are two ways to do pooling. The first way is to take a window within a matrix and pick the max value from within; the first block in the preceding diagram illustrates this approach. The other way is to select a window and pick the max value but, for the next value, instead of moving the window altogether to the new values, slightly slide it either horizontally or vertically to create a partial move effect. The second block in the preceding diagram illustrates the sliding window approach.

After completing all the iteration of a receptive field as well as the pooling, we come to the activation function phase. We will take an example of a ReLU function. In this function, we will tell the resulted matrix that if it contains any negative value, then change that value to become zero.

In the classification phase, you flatten all the outputs that have been previously gathered into a small dimensional vector and then apply a fully connected layer to it. Then, we apply a probability Softmax function in the end and take the highest value in comparison with all the learned images and features. For example, from with the output values, the highest value is 81%, which belongs to a smiley face image; this means that the input image is most likely to be similar to a smiley face.

Let us now explore more about deep learning and how we can quickly start creating our own deep neural networks, using the libraries and frameworks that are already available on the market.

Deep learning using TensorFlow

Nowadays, deep learning is a very hot topic. We have already established an understanding about what deep learning is in the previous chapter. Before proceeding further, let's recap a little bit. Deep learning is a branch of machine learning that is based on neural networks similar to a human brain; more specifically, a neural network with a number of hidden layers is called a deep neural network or deep learning. The more hidden layers, the deeper the network and learning will be.

There are many open source libraries and frameworks available that can handle the complex computation involved in designing a neural network. Some of them are listed as follows:

- TensorFlow
- Caffe
- Theano
- Keras
- Apache MXNet
- Torch
- Microsoft Cognitive Toolkit
- Pytorch
- Deeplearning4j
- Lasagne

We will be going through the TensorFlow framework in the subsequent section.

TensorFlow

TensorFlow is an open source framework designed to handle the complexity involved in designing a neural network. This complexity includes numerical computation with data flow graphs to run a deep neural network. TensorFlow was originally designed by the researchers and engineers of the Google Brains Team in the Google Machine Intelligence Research Organization. It was then released under an Apache open source license in 2015, as architecturally it is more general purpose in use.

It can run on multiple CPUs and GPUs. It is designed to run across almost all platforms, including desktop (Linux, Windows, macOS) and mobile devices (Android and iOS).

Let's proceed by installing TensorFlow in our system.

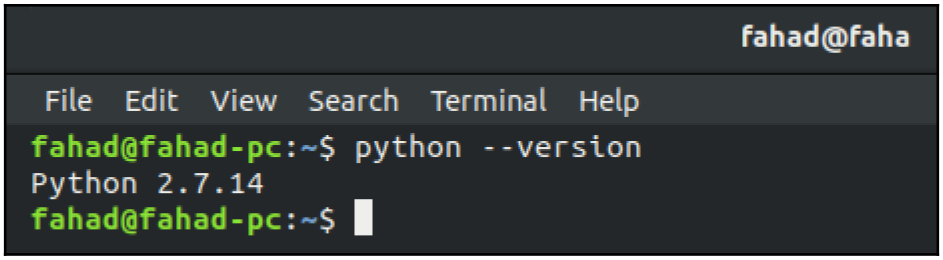
Installation

There are multiple ways of installing TensorFlow in your system. It can be installed in a virtual environment or as a Docker image. We will be going through the steps of having it installed locally, or in the native environment.

The first step is to check whether `python` is already installed on your computer by issuing the following command in your terminal:

```
$ python --version
```

If it is already installed, it will display an output omitting the version number of the installed Python, as shown in the following screenshot:



```
fahad@fahad-pc:~$ python --version
Python 2.7.14
fahad@fahad-pc:~$
```

Installed Python version

Otherwise, issue the following command in order to install it on your computer:

```
$ sudo apt-get install python
```

Once `python` is installed on your computer, it's time to install the package manager for Python, which is called `pip`. As the name states, it is used to install and manage packages for `python`.

If you have `python` version 2.7.x on your machine, execute the following command:

```
$ sudo apt-get install python-pip python-dev
```

If you have `python` version 3.x on your machine, execute the following command:

```
$ sudo apt-get install python3-pip python3-dev
```

Once `pip` is installed, it's now time to install TensorFlow. If you have Python version 2.7.x on your machine, execute the following command:

```
$ sudo pip install tensorflow
```

If you have Python version 3.x on your machine, execute the following command:

```
$ sudo pip3 install tensorflow
```

If your computer has the GPU version, then there is a version of TensorFlow that supports and runs on GPU for fast computation and processing. To install the GPU version of TensorFlow, execute the following command:

```
# For Python 2.7.x
$ sudo pip install tensorflow-gpu

# For Python 3.x
$ sudo pip3 install tensorflow-gpu
```

Now that we have successfully installed TensorFlow in our computer, it's time to execute a small program to test the installation and see whether it is installed correctly or not.

TensorFlow program

We will now run a small program in a Python interactive shell terminal. The purpose of this program is just to import the main package of TensorFlow and run a command using it. If it is able to import the package and run the program successfully, it means our development environment for TensorFlow is ready.

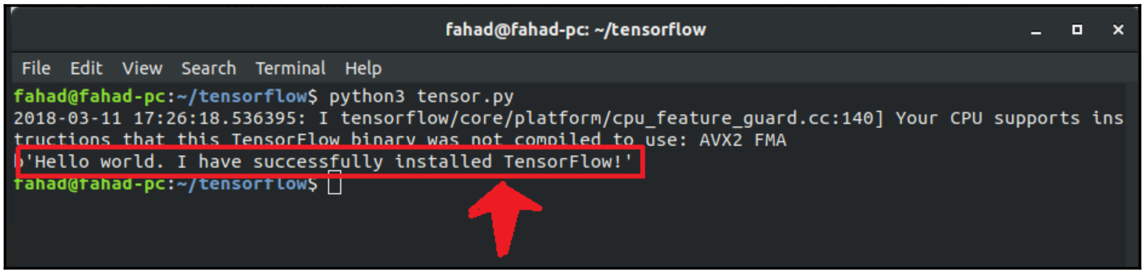
The first step is to copy the following code in a file and save it as `tensor.py`. The complete path in my case is `/home/fahad/tensorflow/tensor.py`:

```
import tensorflow as tf

hello = tf.constant('Hello world. I have successfully installed
TensorFlow!')

sess = tf.Session()
print(sess.run(hello))
```

Once you save it, navigate to the saved path from within your terminal and execute the following command. In my case, I have done the TensorFlow installation for Python 3.x:

```
$ python3 tensor.py
```

You will see an output similar to the one shown in the following screenshot:


fahad@fahad-pc: ~/tensorflow
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/tensorflow$ python3 tensor.py
2018-03-11 17:26:18.536395: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports ins
tructions that this TensorFlow binary was not compiled to use: AVX2 FMA
b'Hello world. I have successfully installed TensorFlow!'
fahad@fahad-pc:~/tensorflow$
```

TensorFlow program output to validate the installation

As highlighted in the preceding image, we are able to get an output from our hello world program of TensorFlow. This means that we have successfully installed it on our computer.

Uninstalling TensorFlow

If at any point of time you need to uninstall TensorFlow from your computer, just execute the following command, if you installed it for Python 2.7.x:

```
$ sudo pip uninstall tensorflow
```

If you installed TensorFlow for Python 3.x, execute the following command:

```
$ sudo pip3 uninstall tensorflow
```

Now that we have done the installation of TensorFlow and run a basic program using it, in the subsequent section, we will learn how to visualize a TensorFlow program to help to view the flow of the program for easy understanding and to identify the issues in it. This will be done using a TensorFlow module named TensorBoard.

TensorBoard

TensorBoard is a very useful tool if you are using TensorFlow on your machine and for deep learning programs. It displays a graphical representation of different variables and layers involved in the overall execution of your program for easy understanding. Your program can be complex and if you encounter any program in it, it will be very difficult to debug - for example, which layer is generating an error? Or, how is the execution actually happening?

TensorBoard is a part of TensorFlow and is installed by default during the installation of TensorFlow. You can even see it being installed during the installation of TensorFlow when it tries to download the TensorBoard package and installs it.

We will now create a TensorFlow program to solve a mathematical equation, and will visualize it in TensorBoard.

Program

We have the following mathematical equation:

$$Result = (A * B) + (C/B) * A - C$$

The following code has been implemented to calculate the result of the given equation using the TensorFlow framework in Python:

```
import tensorflow as tf

# Define the required placeholder
A = tf.placeholder(tf.float32, name="a")
B = tf.placeholder(tf.float32, name="b")
C = tf.placeholder(tf.float32, name="c")

# Equation to solve
```

```
calculation = (A * B) + (C / B) * A - C

# Starting TensorFlow session to calculate result
with tf.Session() as session:
    result = session.run(calculation, feed_dict= {A: 1, B: 2, C: 3})
    # Output the session in a log directory
    writer =
tf.summary.FileWriter('/home/fahad/tensorflow/log/calculation',
session.graph)

    # Output the result in terminal
    print (result)
```

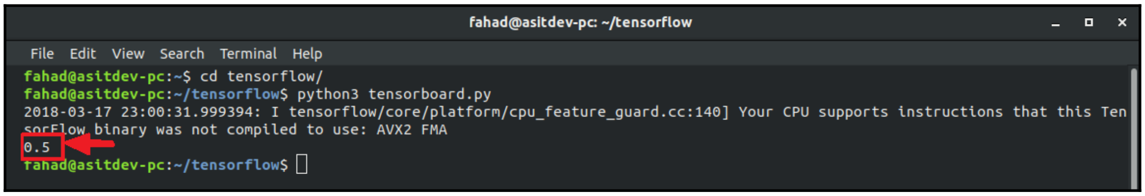
In order to get the visual representation, the following is the most important line in our code, as it places a generated log file so that TensorBoard can read the related data from it and generate a graph accordingly. Please note that the path mentioned here will be used later while starting TensorBoard:

```
tf.summary.FileWriter('/home/fahad/tensorflow/log/calculation',
session.graph)
```

The next step is to execute the following command in order to compile the code and run it:

```
$ python3 tensorboard.py
```

The output generated using this command is shown in the following figure. As highlighted, it computes the result `0.5` of the given equation in the code using the values A=1, B=2, and C=3:



```
fahad@asitdev-pc:~$ cd tensorflow/
fahad@asitdev-pc:~/tensorflow$ python3 tensorboard.py
2018-03-17 23:00:31.999394: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
0.5
fahad@asitdev-pc:~/tensorflow$
```

Execution output of TensorFlow program

Launching TensorBoard

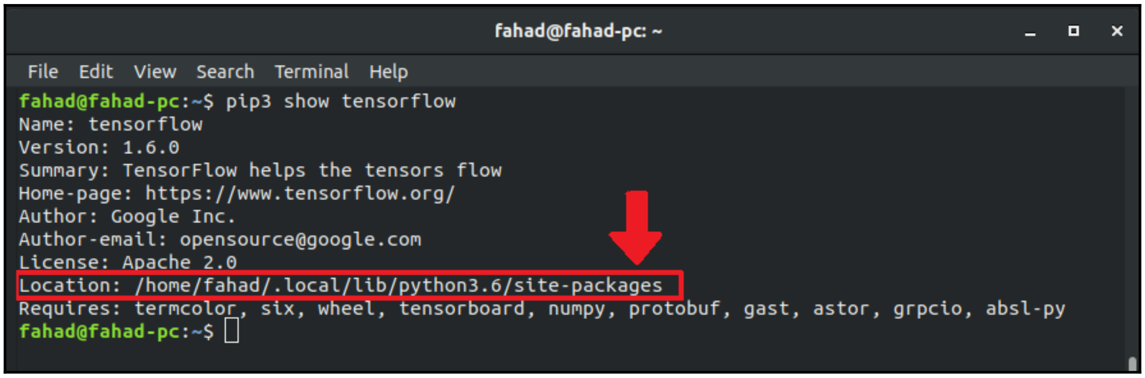
Now that we have successfully executed our code, the program should have generated the related log file in the location mentioned in the code. You can verify it by browsing the folder. If the log folder contains the file after the execution of the program, it means we are all set to proceed to the next step.

The next step is to find out the location of the `site-packages` directory where **Python package manager** (**pip**) installs all the related modules and frameworks. Execute the following command to get the location of the directory where TensorFlow is installed:

```
$ pip3 show tensorflow
```

If you are using Python version 2.7.x, then use pip instead of `pip3`.

This command will generate an output similar to the following screenshot. As highlighted with a red box, it displays the location where TensorFlow is installed:



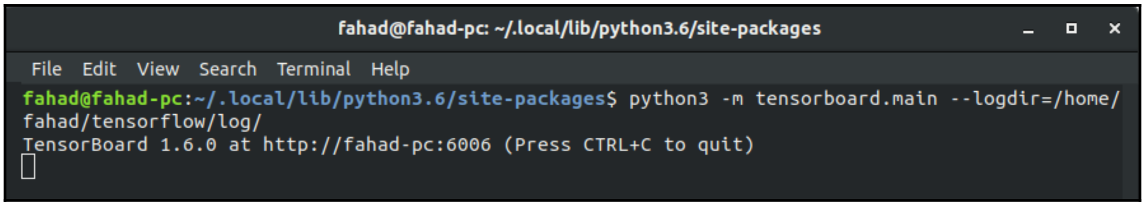
```
fahad@fahad-pc:~$ pip3 show tensorflow
Name: tensorflow
Version: 1.6.0
Summary: TensorFlow helps the tensors flow
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: opensource@google.com
License: Apache 2.0
Location: /home/fahad/.local/lib/python3.6/site-packages
Requires: termcolor, six, wheel, tensorboard, numpy, protobuf, gast, astor, grpcio, absl-py
fahad@fahad-pc:~$
```

Identifying TensorFlow installation path

Move to this directory and issue the following command to start TensorBoard. It requires a `logdir` parameter to read the program output from it. In my case, I have defined the log directory location as `/home/fahad/tensorflow/log`:

```
$ python3 -m tensorboard.main --logdir=/home/fahad/tensorflow/log
```

This will display an output similar to the following screenshot. It also contains the URL from where you can access the TensorBoard by pointing your browser to this location:


fahad@fahad-pc: ~/.local/lib/python3.6/site-packages
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/.local/lib/python3.6/site-packages$ python3 -m tensorboard.main --logdir=/home/
fahad/tensorflow/log/
TensorBoard 1.6.0 at http://fahad-pc:6006 (Press CTRL+C to quit)
```

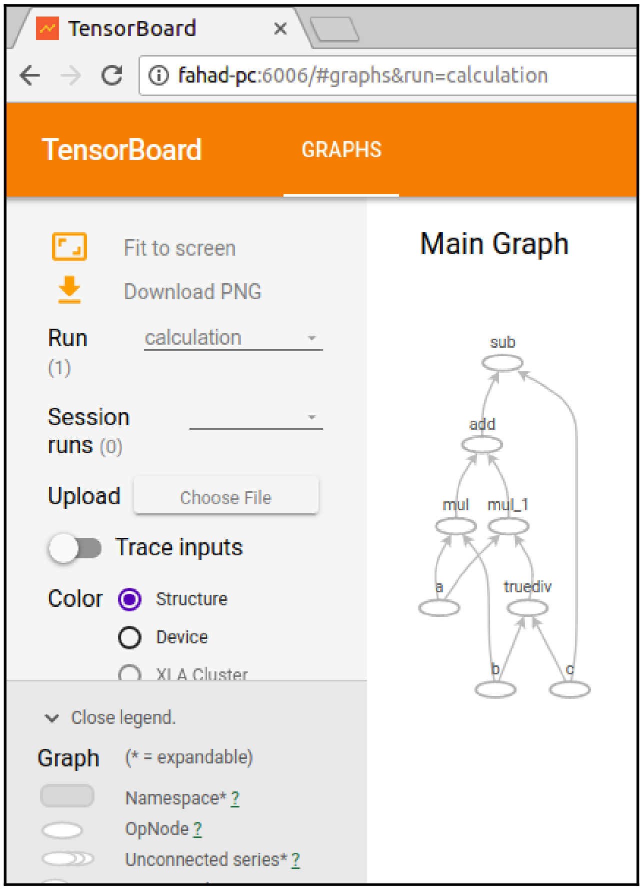
Starting TensorBoard

By default, it publishes TensorBoard to port `6006`, but if it is in use by any other program in your computer, you can pass a desired port number along with the command. Your command will then look as follows:

```
$ python3 -m tensorboard.main --logdir=/home/fahad/tensorflow/log --
port=7007
```

TensorBoard graph

Now that we have successfully started our TensorBoard, point your browser to the URL mentioned in the resulting output. It will display a page similar to the following screenshot:



Graph generated by the TensorBoard using the log file generated by TensorFlow program

It is displaying a graphical representation of the many layers/steps it has to go through to solve the provided equation, what the initial inputs were, and how it reaches the desired output. Now, imagine the case of a more complex program where there are many more layers and many more inputs and outputs; this tool will be of much use to understand the flow and debug any issues encountered.

We will now proceed to the next topic of this chapter, in which we will learn how to detect different objects with an image using the YOLO library.

Object detection using YOLO

YOLO stands for **you only look once**. It's an object detection API. It uses a convolutional neural network to detect objects in an image. In the traditional detection system, it applies a model into different parts of the captured image to detect whether the same model is present. YOLO uses a neural network for this purpose. It divides an image into chunks and then applies edge detection to identify the object and calculate the probability of having an object in that region. It then picks up from there and detects the object from within its pre-trained data.

Let's get our hands dirty with YOLO by starting with its installation process.

Installation

The full code base of YOLO is available in the GitHub repository for everyone to download. It is allowed to be used by everyone freely. To clone the `git` repository, make sure that `git` is installed on your computer. Issue the following command to install `git`:

```
$ sudo apt-get install git
```

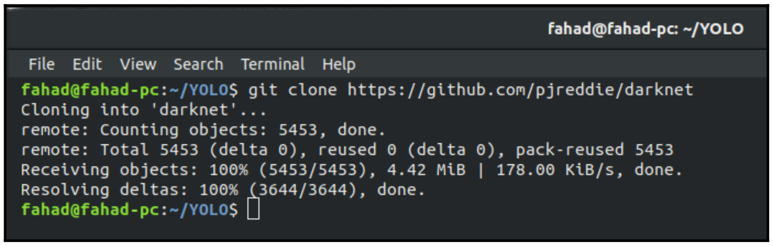
This command will install `git` on your computer, if it's not already installed. The next step is to create a directory in your home folder called `YOLO`:

```
$ mkdir ~/YOLO/
$ cd ~/YOLO/
```

The next step is to `clone` the `git` repository by issuing the following command:

```
$ git clone https://github.com/pjreddie/darknet
```

It will display an output similar to the following screenshot:



```
fahad@fahad-pc:~/YOLO$ git clone https://github.com/pjreddie/darknet
Cloning into 'darknet'...
remote: Counting objects: 5453, done.
remote: Total 5453 (delta 0), reused 0 (delta 0), pack-reused 5453
Receiving objects: 100% (5453/5453), 4.42 MiB | 178.00 KiB/s, done.
Resolving deltas: 100% (3644/3644), done.
fahad@fahad-pc:~/YOLO$
```

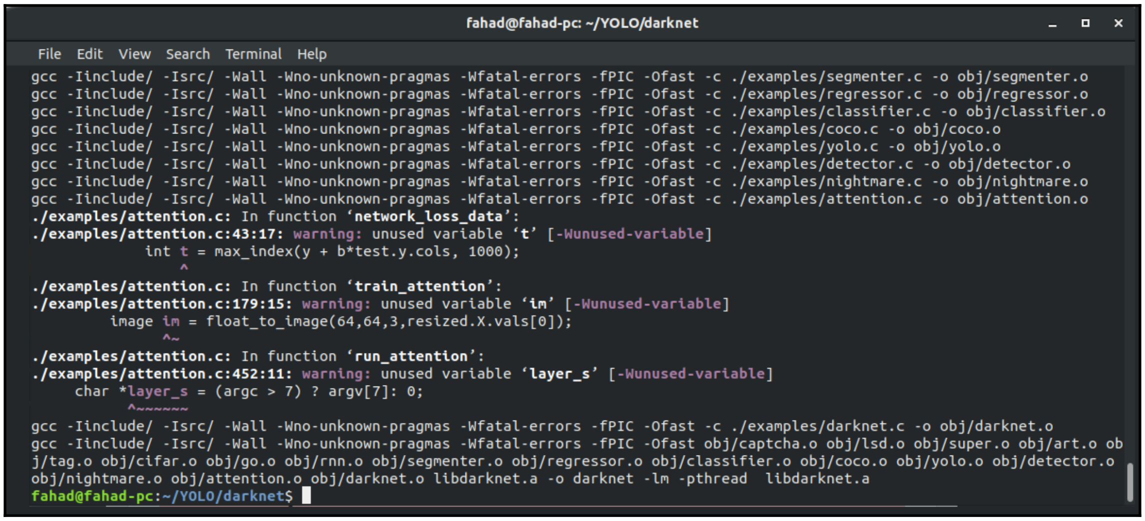
Cloning the Git Repository of YOLO Darknet

Compiling YOLO library

Once the download finishes, it's time to browse into the darknet directory and compile the code by executing the following command:

```
$ cd darknet
$ make
```

The `make` command will compile the code. If it finishes with only a warning, without any errors, as shown in the following figure, it means that YOLO darknet has been compiled successfully and is ready to use. If it displays any errors and ends unexpectedly, try to redo the previously mentioned steps:


fahad@fahad-pc: ~/YOLO/darknet
File Edit View Search Terminal Help

```
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/segmenter.c -o obj/segmenter.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/regressor.c -o obj/regressor.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/classifier.c -o obj/classifier.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/coco.c -o obj/coco.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/yolo.c -o obj/yolo.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/detector.c -o obj/detector.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/nightmare.c -o obj/nightmare.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/attention.c -o obj/attention.o
./examples/attention.c: In function ‘network_loss_data’:
./examples/attention.c:43:17: warning: unused variable ‘t’ [-Wunused-variable]
             int t = max_index(y + b*test.y.cols, 1000);
                 ^
./examples/attention.c: In function ‘train_attention’:
./examples/attention.c:179:15: warning: unused variable ‘im’ [-Wunused-variable]
         image im = float_to_image(64,64,3,resized.X.vals[0]);
               ^~
./examples/attention.c: In function ‘run_attention’:
./examples/attention.c:452:11: warning: unused variable ‘layer_s’ [-Wunused-variable]
     char *layer_s = (argc > 7) ? argv[7]: 0;
           ^~~~~~~
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/darknet.c -o obj/darknet.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast obj/captcha.o obj/lsd.o obj/super.o obj/art.o ob
j/tag.o obj/cifar.o obj/go.o obj/rnn.o obj/segmenter.o obj/regressor.o obj/classifier.o obj/coco.o obj/yolo.o obj/detector.o
obj/nightmare.o obj/attention.o obj/darknet.o libdarknet.a -o darknet -lm -pthread  libdarknet.a
fahad@fahad-pc:~/YOLO/darknet$
```

Output of the Make command executed in the Darknet folder

Trained weights

The YOLO creators have provided a trained model weight file. This weight file contains data that is trained using thousands of images and objects. You can create your own weight file by training a model, but that's a story for another time. For now, we will use the `yolo.weights` file to detect objects in an image. The weight file is approximately 258 MB in size. You can download it by executing the following command. Make sure to execute this command in the darknet directory:

```
$ wget https://pjreddie.com/media/files/yolo.weights
```

Detecting objects in an image

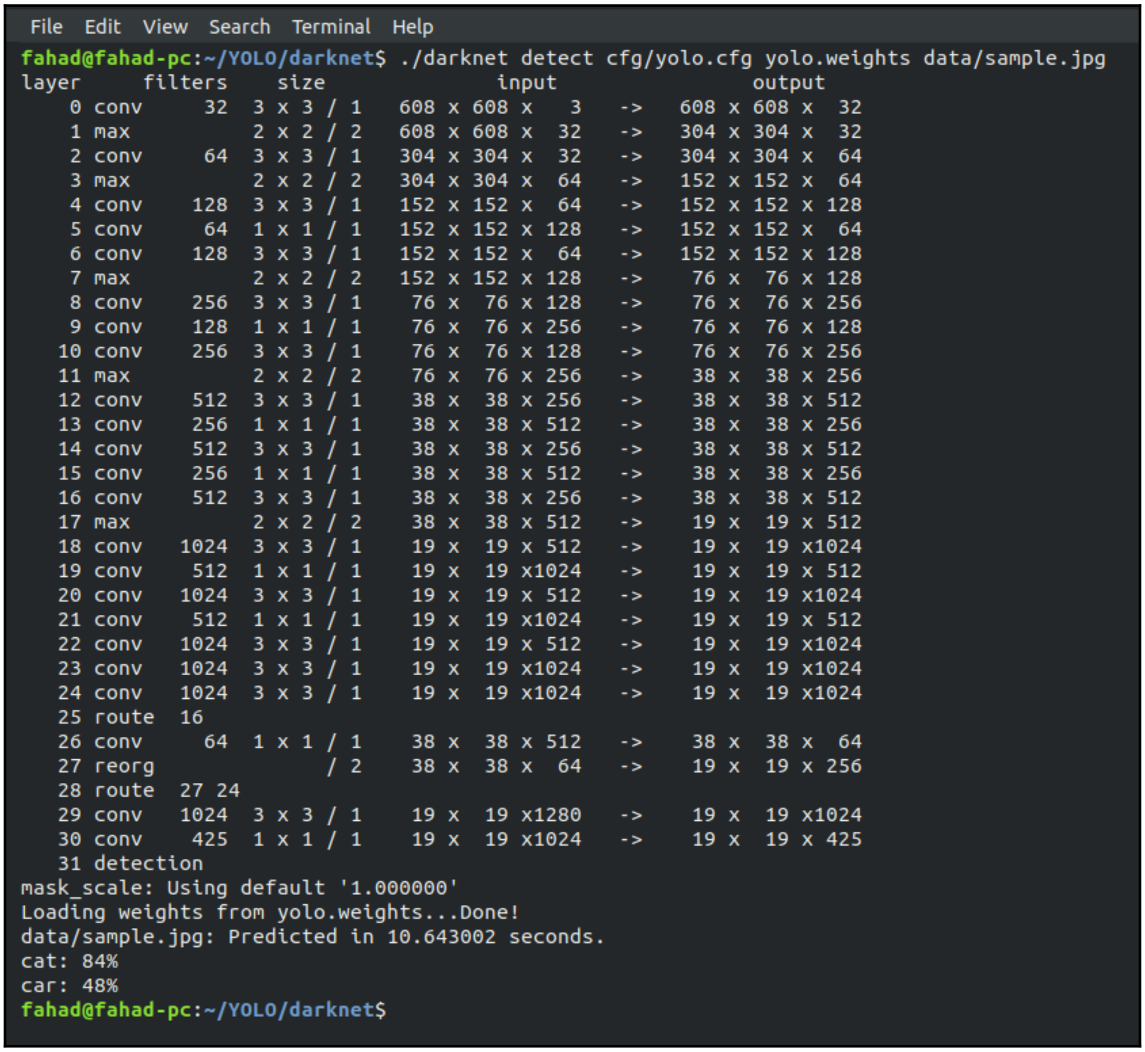
We will be using the following image as a sample to detect objects in it. Place this image in the `data` directory within the `darknet` folder. Placing the sample image is not mandatory; we just placed it here for the sake of simplicity. You can place the image anywhere you want:

Sample image to detect objects using YOLO darknet

Now that we have successfully compiled `darknet`, downloaded the `weights`, and have the sample image ready, it's time to run it and detect objects from our sample image. To do so, execute the following command:

```
$ ./darknet detect cfg/yolo.cfg yolo.weights data/sample.jpg
```

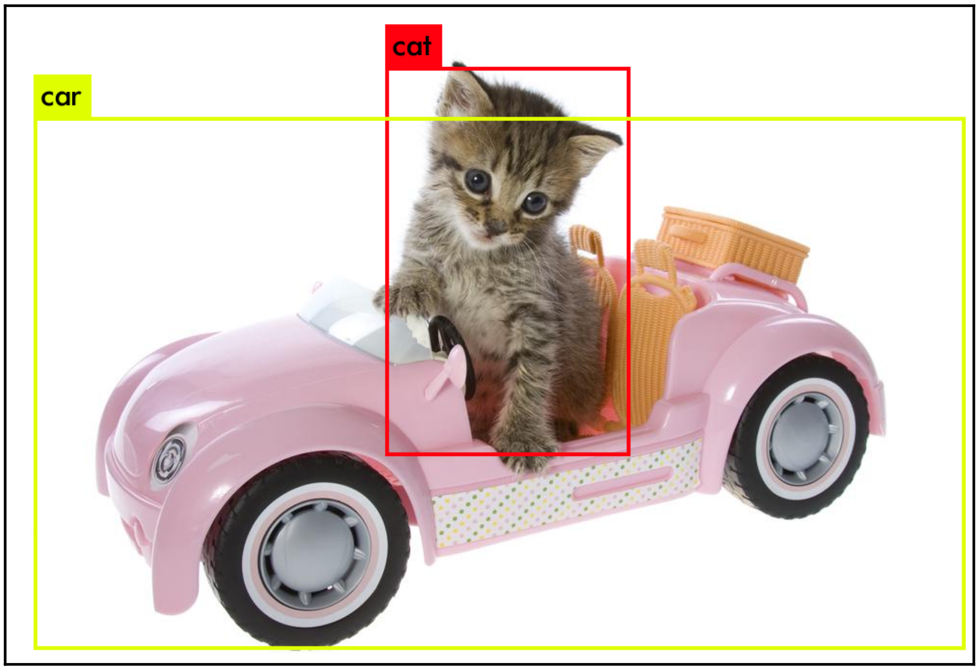
It will display an output similar to the following screenshot:



```
fahad@fahad-pc:~/YOLO/darknet$ ./darknet detect cfg/yolo.cfg yolo.weights data/sample.jpg
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   608 x 608 x   3   ->   608 x 608 x  32
    1 max          2 x 2 / 2   608 x 608 x  32   ->   304 x 304 x  32
    2 conv     64  3 x 3 / 1   304 x 304 x  32   ->   304 x 304 x  64
    3 max          2 x 2 / 2   304 x 304 x  64   ->   152 x 152 x  64
    4 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128
    5 conv     64  1 x 1 / 1   152 x 152 x 128   ->   152 x 152 x  64
    6 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128
    7 max          2 x 2 / 2   152 x 152 x 128   ->    76 x  76 x 128
    8 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256
    9 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128
   10 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256
   11 max          2 x 2 / 2    76 x  76 x 256   ->    38 x  38 x 256
   12 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512
   13 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256
   14 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512
   15 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256
   16 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512
   17 max          2 x 2 / 2    38 x  38 x 512   ->    19 x  19 x 512
   18 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024
   19 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512
   20 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024
   21 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512
   22 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024
   23 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024
   24 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024
   25 route  16
   26 conv     64  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x  64
   27 reorg              / 2    38 x  38 x  64   ->    19 x  19 x 256
   28 route  27 24
   29 conv   1024  3 x 3 / 1    19 x  19 x1280   ->    19 x  19 x1024
   30 conv    425  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 425
   31 detection
mask_scale: Using default '1.000000'
Loading weights from yolo.weights...Done!
data/sample.jpg: Predicted in 10.643002 seconds.
cat: 84%
car: 48%
fahad@fahad-pc:~/YOLO/darknet$
```

YOLO darknet execution result on our sample image

As you can see, at the end of the execution, it displayed its result. It took `10.64` seconds to predict that the provided image has an `84%` chance of containing an image of a cat. Along with a cat, there is a `48%` chance that it also contains an image of a car. By default, it also creates the end result in a form of a predicted result where it draws a box on each and every object it found in the provided image. It then saves the resulting image in the root of the darknet directory. In our case, the complete path of the predicted image is `~/YOLO/darknet/predictions.png`. The predicted image is shown as follows:



Output generated by YOLO darknet

After analyzing this image, YOLO predicted the objects within it very well, even though it's output only has a `48%` chance of containing a car.

Summary

In this chapter, we discussed what artificial intelligence is, starting from when it was a theory and considering how it evolved from there. We also discussed how all the rapid advancements in technology are changing the world around it. We then discussed the most important concept in AI, CNNs, in which we try to mimic the functionality of a human brain and how it pursues any given task.

We then listed down some libraries that are very useful in terms of designing neural networks, especially deep neural networks. We then got into some details about the most used library among them, which is developed by Google and named TensorFlow. After an introduction about it, we proceeded further by setting up its environment so that we could use it to develop our own neural network program. We then learned about the TensorBoard tool, namely how to configure and use it. It comes with TensorFlow. The purpose of it is to display the execution steps of TensorFlow programs in a graphical form.

We then moved our discussion to object detection in an image using Neural Networks. We looked at a library called YOLO, which works on the principle of neural networks and generates very fast results once trained. We went through its installation steps, covering downloading the project code, configuring it, and compiling it. We then used the trained weights data that is provided by the YOLO team to get you started. We used a sample image and detected the objects in it; the predictions were impressive.

In the next chapter, we will learn how to use Elasticsearch, which is an open source tool. It is an enterprise-grade search engine specifically designed for use with big data.

14
Elasticsearch

Elasticsearch is a full text-based search and analytics engine. It is a highly scalable and distributed system, specifically designed to work efficiently and quickly with big data systems, where one of its main use cases is log analysis. It is capable of performing advanced and complex searches, and almost real-time processing for advanced analytics and operational intelligence.

Elasticsearch is written in Java and is based on Apache Lucene. It was first released in 2010 and it immediately gained popularity because of its flexible data structure, scalable architecture, and very fast response time. Elasticsearch is based on a JSON document with a schema-free structure, making adoption easy and hassle-free. It is one of the top ranking search engines of enterprise grade. You can write its client in any programming language; Elasticsearch officially works with Java, .NET, PHP, Python, Perl, and so on.

Elasticsearch mainly interacts using a REST API. It gets data in the form of JSON documents with all of the required parameters, and provides its response in a similar fashion. We will look at this in more detail when we deep-dive into practical tasks and execute API calls.

In this chapter, we will discuss the following topics:

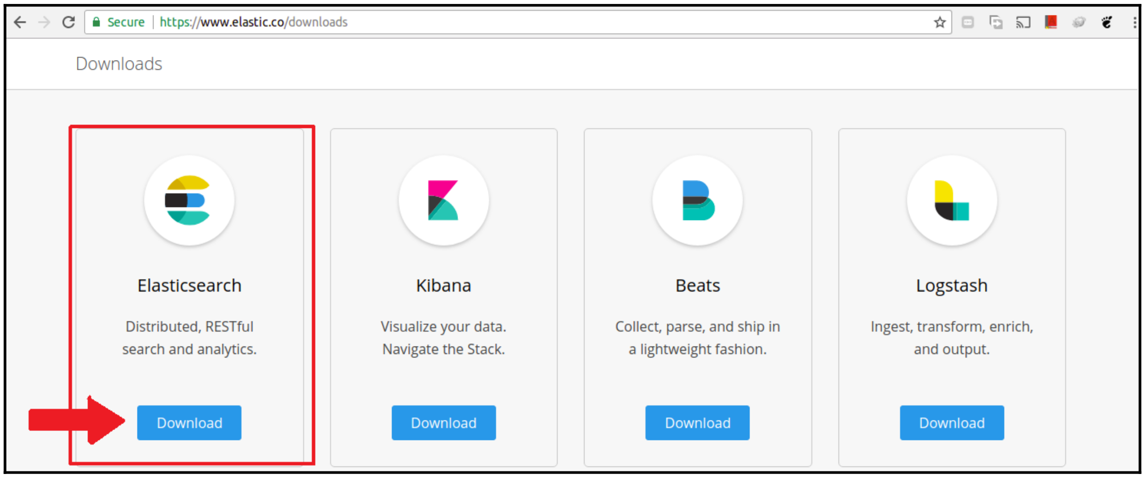
- Installing Elasticsearch
- Kibana
- Installing Kibana
- Securing Elasticsearch
- Understanding queries in Elasticsearch

Let's now move on and set up Elasticsearch in our environment for the first time.

Installing Elasticsearch

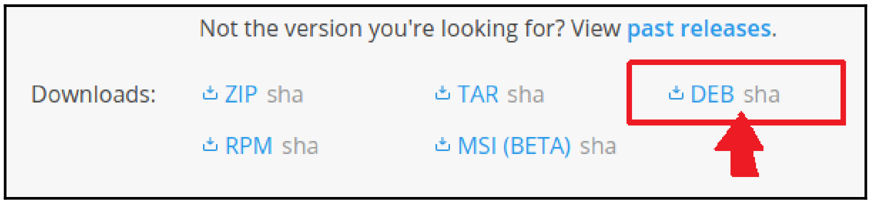
Let's first start with the installation procedure and set up Elasticsearch. These are the steps required to get started:

1. Point your browser to the URL `https://www.elastic.co/downloads`.
2. Click on the **Download** button in the Elasticsearch box to open up the web page and select the required operating system, as shown in the following screenshot:


Secure | https://www.elastic.co/downloads
Downloads
Elasticsearch
Distributed, RESTful search and analytics.
Download
Kibana
Visualize your data. Navigate the Stack.
Download
Beats
Collect, parse, and ship in a lightweight fashion.
Download
Logstash
Ingest, transform, enrich, and output.
Download

Elasticsearch download button

3. On the next page, select the desired operating system and download the related package. In our case, click on the **DEB** link to download the Debian package to install Elasticsearch in Ubuntu, as shown in the following screenshot:

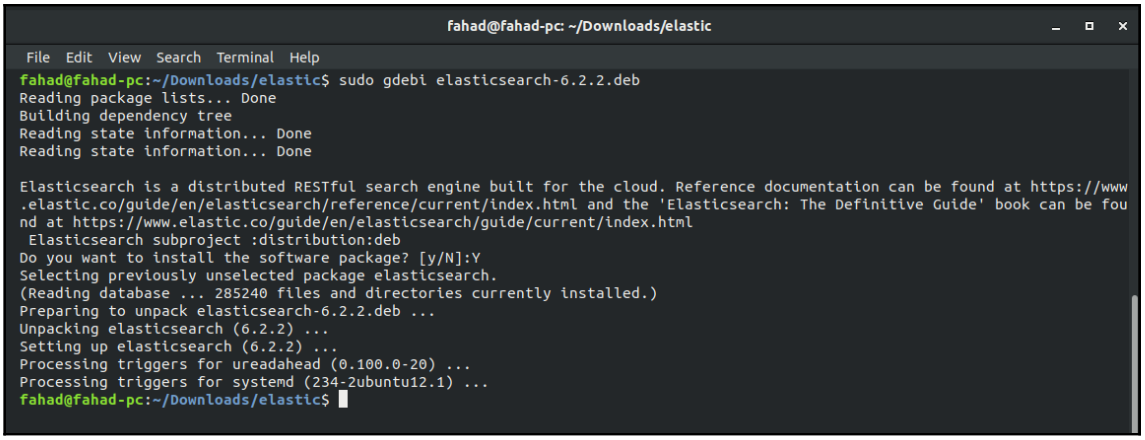


Selecting the desired Elasticsearch package

4. Once the download is complete, double click on the file to open the install function, or execute the following command to install it:

```
$ sudo gdebi elasticsearch-6.2.2.deb
```

The following is a snapshot of the output that is produced after the preceding command is run:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo gdebi elasticsearch-6.2.2.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading state information... Done

Elasticsearch is a distributed RESTful search engine built for the cloud. Reference documentation can be found at https://www
.elastic.co/guide/en/elasticsearch/reference/current/index.html and the 'Elasticsearch: The Definitive Guide' book can be fou
nd at https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html
 Elasticsearch subproject :distribution:deb
Do you want to install the software package? [y/N]:Y
Selecting previously unselected package elasticsearch.
(Reading database ... 285240 files and directories currently installed.)
Preparing to unpack elasticsearch-6.2.2.deb ...
Unpacking elasticsearch (6.2.2) ...
Setting up elasticsearch (6.2.2) ...
Processing triggers for ureadahead (0.100.0-20) ...
Processing triggers for systemd (234-2ubuntu12.1) ...
fahad@fahad-pc:~/Downloads/elastic$
```

Execution of the installation command for Elasticsearch

TIP

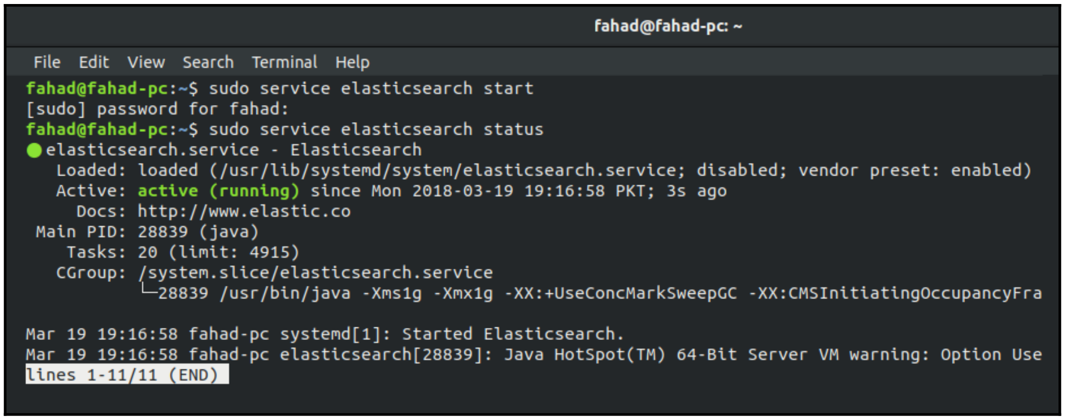
`gdebi` is a Debian package installer that is used to identify if any dependent package is also required before installation. Use `sudo apt install gdebi` to install it.

Starting the Elasticsearch server

We have now successfully installed Elasticsearch. It's now time to started! So, execute the following command:

```
$ sudo service elasticsearch start
```

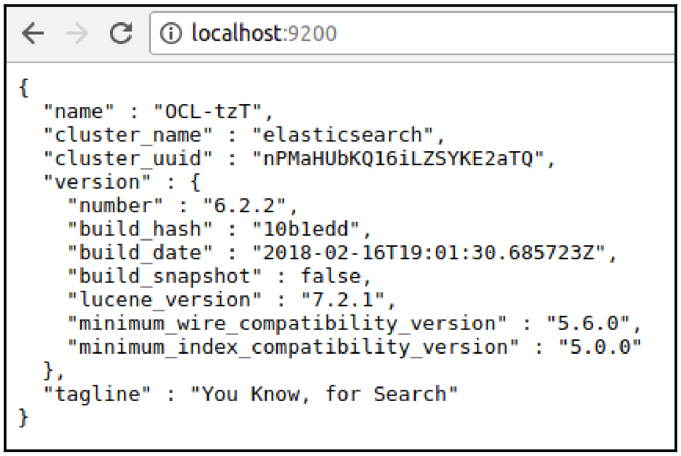
The following screenshot shows the output of the `start` command:



```
fahad@fahad-pc:~$ sudo service elasticsearch start
[sudo] password for fahad:
fahad@fahad-pc:~$ sudo service elasticsearch status
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2018-03-19 19:16:58 PKT; 3s ago
     Docs: http://www.elastic.co
 Main PID: 28839 (java)
    Tasks: 20 (limit: 4915)
   CGroup: /system.slice/elasticsearch.service
           └─28839 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFra

Mar 19 19:16:58 fahad-pc systemd[1]: Started Elasticsearch.
Mar 19 19:16:58 fahad-pc elasticsearch[28839]: Java HotSpot(TM) 64-Bit Server VM warning: Option Use
lines 1-11/11 (END)
```

Start and status of the Elasticsearch service

Once Elasticsearch has successfully started, point your browser to `http://localhost:9200` to view the output in a browser, as shown in the following screenshot:



```
{
  "name" : "OCL-tzT",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "nPMaHUbKQ16iLZSYKE2aTQ",
  "version" : {
    "number" : "6.2.2",
    "build_hash" : "10b1edd",
    "build_date" : "2018-02-16T19:01:30.685723Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

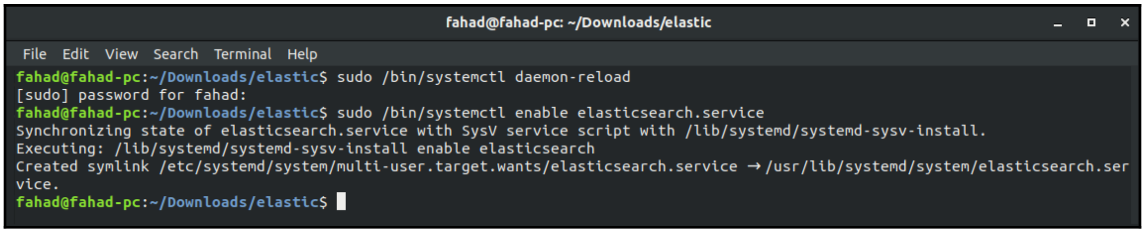
Elasticsearch running confirmation on the browser

Auto starting the Elasticsearch service

Normally, in a production environment, it is essential to start all of the required services whenever the server itself starts. To automatically start Elasticsearch along with your server, execute the following command:

```
$ sudo /bin/systemctl daemon-reload
$ sudo /bin/systemctl enable elasticsearch.service
```

The following screenshot illustrates the output of the preceding command:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo /bin/systemctl daemon-reload
[sudo] password for fahad:
fahad@fahad-pc:~/Downloads/elastic$ sudo /bin/systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /usr/lib/systemd/system/elasticsearch.ser
vice.
fahad@fahad-pc:~/Downloads/elastic$
```

Auto-start configuration for the Elasticsearch command output

Stopping the Elasticsearch server

At any point in time, if you need to stop the Elasticsearch service, execute the following command:

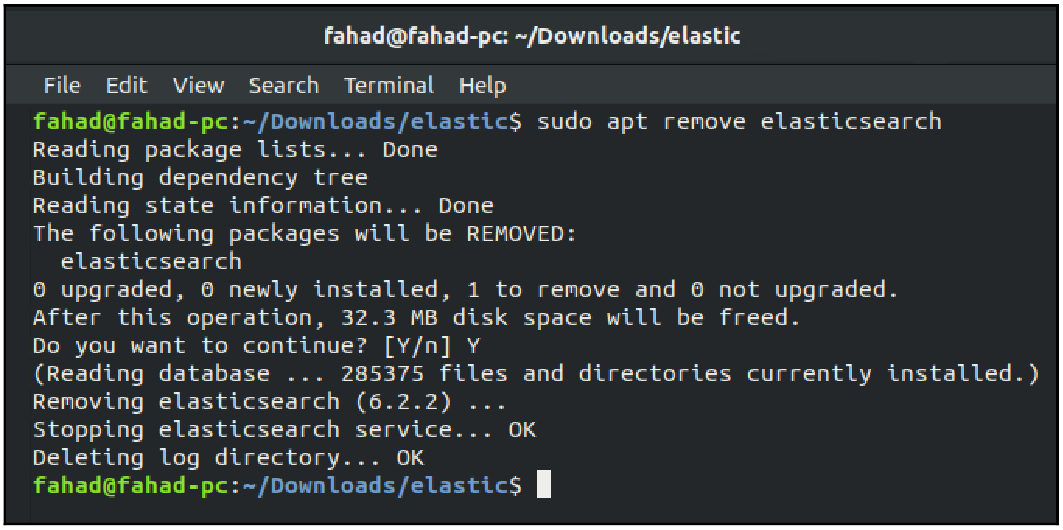
```
$ sudo service elasticsearch stop
```

Uninstalling Elasticsearch

If, for any reason, you need to uninstall Elasticsearch from your computer or server, execute the following command:

```
$ sudo apt remove elasticsearch
```

The following screenshot displays the preceding command's execution and the output:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo apt remove elasticsearch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  elasticsearch
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 32.3 MB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 285375 files and directories currently installed.)
Removing elasticsearch (6.2.2) ...
Stopping elasticsearch service... OK
Deleting log directory... OK
fahad@fahad-pc:~/Downloads/elastic$
```

Uninstalling Elasticsearch, command and output

Let's now move our discussion on to a tool that is used to visually interact with Elasticsearch – Kibana.

Kibana

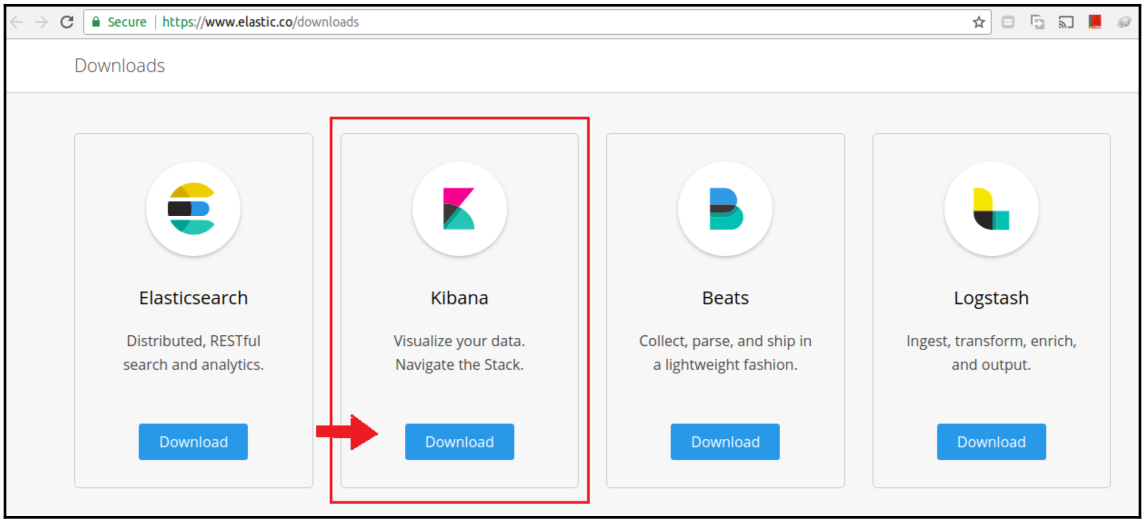
Kibana was developed by the Elasticsearch team and was released as an open source project. It is very easy to use, yet is powerful enough to interact with Elasticsearch and perform analysis on demand. You can easily run a Kibana query by calling the Elasticsearch API. We have already learned that Elasticsearch interacts using REST API calls, so the query execution is a little different to what we use with a traditional SQL. In this case, Kibana provides a very simple environment for performing different actions by calling Elasticsearch API.

In Kibana, you can also create multiple dashboards and design different types of charts to visualize data – it's a well-known saying that a picture is worth a thousand words. Let's now move on to the installation of Kibana; we will explore its functionalities and how to use it later.

Installation

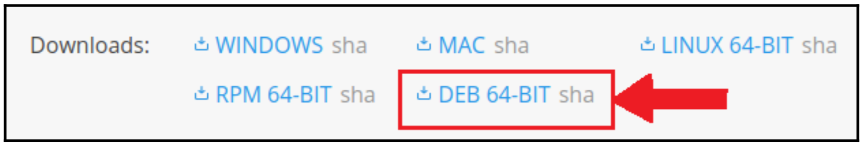
Let's now install Kibana. Follow the proceeding steps to get started:

1. Point your browser to the URL `https://www.elastic.co/downloads`.
2. Click on the **Download** button in the Kibana box to open up the web page and select the required operating system, as shown in the following screenshot:


Secure
https://www.elastic.co/downloads
Downloads
Elasticsearch
Distributed, RESTful search and analytics.
Download
Kibana
Visualize your data. Navigate the Stack.
Download
Beats
Collect, parse, and ship in a lightweight fashion.
Download
Logstash
Ingest, transform, enrich, and output.
Download

Kibana download page

3. On the next page, select the desired operating system to download the related package. In our case, click on the **DEB** link to download the Debian package to install Kibana in Ubuntu, as shown in the following screenshot:

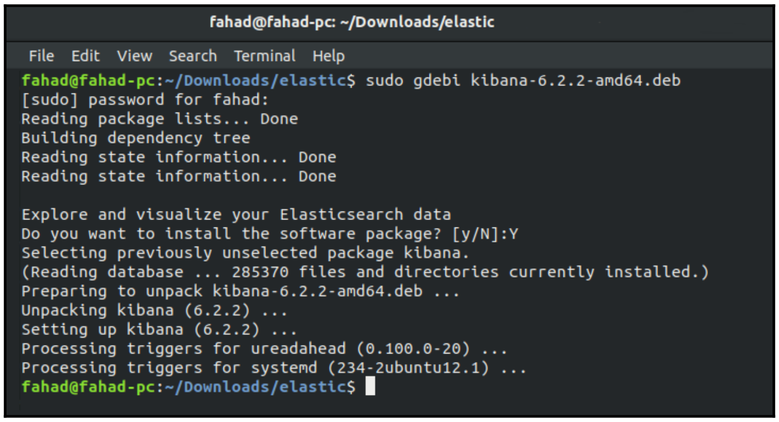


Selecting the desired Kibana package

4. Once the download is complete, double click on the file to install Kibana, or execute the following command:

```
$ sudo gdebi kibana-6.2.2-amd64.deb
```

To install Kibana, execute the preceding command using the command line, as shown in the following screenshot:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo gdebi kibana-6.2.2-amd64.deb
[sudo] password for fahad:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading state information... Done

Explore and visualize your Elasticsearch data
Do you want to install the software package? [y/N]:Y
Selecting previously unselected package kibana.
(Reading database ... 285370 files and directories currently installed.)
Preparing to unpack kibana-6.2.2-amd64.deb ...
Unpacking kibana (6.2.2) ...
Setting up kibana (6.2.2) ...
Processing triggers for ureadahead (0.100.0-20) ...
Processing triggers for systemd (234-2ubuntu12.1) ...
fahad@fahad-pc:~/Downloads/elastic$
```

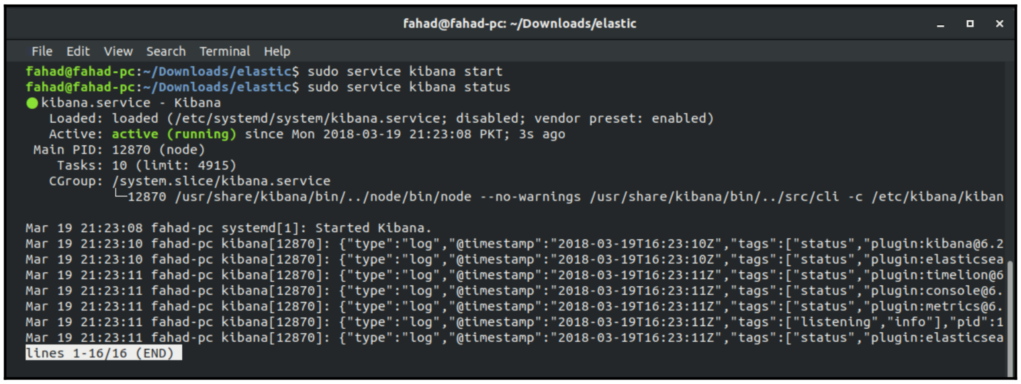
Kibana installation, command and output

Starting Kibana

We have now successfully installed Kibana. It's now time to start the Kibana service with the following command:

```
$ sudo service kibana start
```

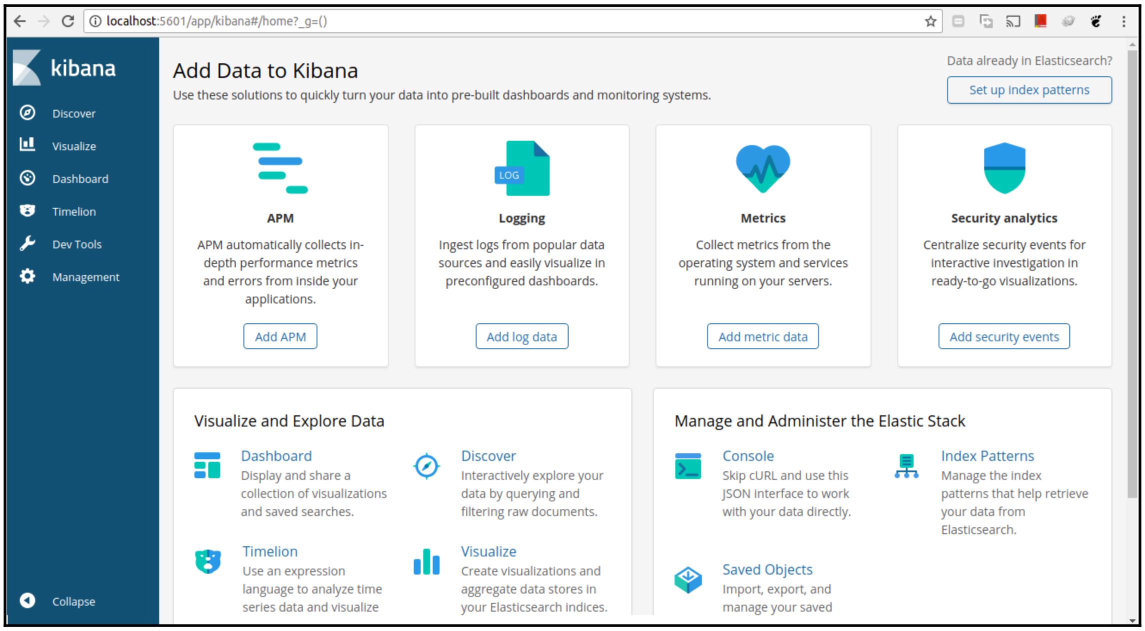
The following screenshot shows the output of the service `start` command:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo service kibana start
fahad@fahad-pc:~/Downloads/elastic$ sudo service kibana status
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2018-03-19 21:23:08 PKT; 3s ago
 Main PID: 12870 (node)
    Tasks: 10 (limit: 4915)
   CGroup: /system.slice/kibana.service
           └─12870 /usr/share/kibana/bin/../node/bin/node --no-warnings /usr/share/kibana/bin/../src/cli -c /etc/kibana/kiban

Mar 19 21:23:08 fahad-pc systemd[1]: Started Kibana.
Mar 19 21:23:10 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:10Z","tags":["status","plugin:kibana@6.2
Mar 19 21:23:10 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:10Z","tags":["status","plugin:elasticsea
Mar 19 21:23:11 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:11Z","tags":["status","plugin:timelion@6
Mar 19 21:23:11 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:11Z","tags":["status","plugin:console@6.
Mar 19 21:23:11 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:11Z","tags":["status","plugin:metrics@6.
Mar 19 21:23:11 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:11Z","tags":["listening","info"],"pid":1
Mar 19 21:23:11 fahad-pc kibana[12870]: {"type":"log","@timestamp":"2018-03-19T16:23:11Z","tags":["status","plugin:elasticsea
lines 1-16/16 (END)
```

Kibana service, start and output

Once the service has successfully started, point your browser to `http://localhost:5601` to open up the Kibana user interface, as shown in the following screenshot:



Add Data to Kibana

Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.

APM automatically collects in-depth performance metrics and errors from inside your applications.

Ingest logs from popular data sources and easily visualize in preconfigured dashboards.

Collect metrics from the operating system and services running on your servers.

Centralize security events for interactive investigation in ready-to-go visualizations.

Visualize and Explore Data

Dashboard: Display and share a collection of visualizations and saved searches.

Discover: Interactively explore your data by querying and filtering raw documents.

Timelion: Use an expression language to analyze time series data and visualize

Visualize: Create visualizations and aggregate data stores in your Elasticsearch indices.

Manage and Administer the Elastic Stack

Console: Skip cURL and use this JSON interface to work with your data directly.

Index Patterns: Manage the index patterns that help retrieve your data from Elasticsearch.

Saved Objects: Import, export, and manage your saved

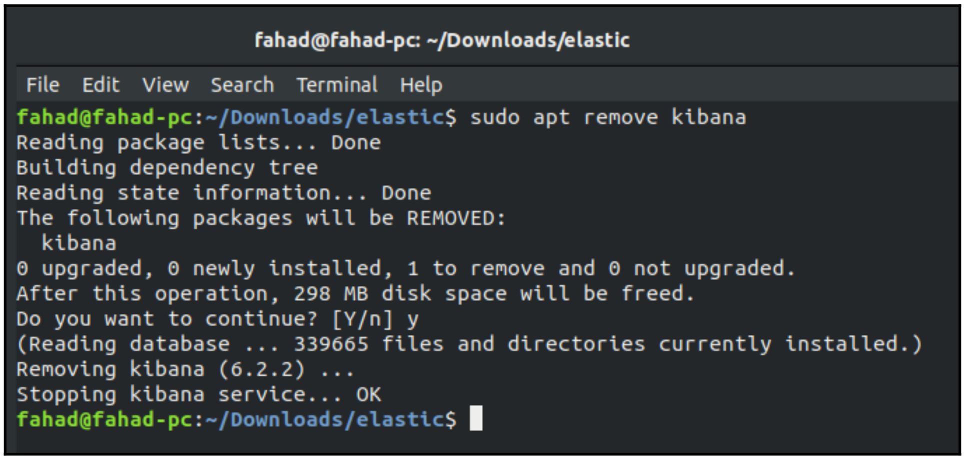
Kibana's User Interface

Uninstalling Kibana

If, at any point in time, you want to uninstall Kibana, execute the following command:

```
$ sudo apt remove kibana
```

The output of the preceding command is shown in the following screenshot:


fahad@fahad-pc: ~/Downloads/elastic
File Edit View Search Terminal Help

```
fahad@fahad-pc:~/Downloads/elastic$ sudo apt remove kibana
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  kibana
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 298 MB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 339665 files and directories currently installed.)
Removing kibana (6.2.2) ...
Stopping kibana service... OK
fahad@fahad-pc:~/Downloads/elastic$
```

Uninstalling Kibana, command and output

We now know what Elasticsearch is and the steps required to install it on your computer. We have also successfully installed Kibana, which helps us to interact with Elasticsearch. Let's now move on to setting up security.

Security

It is always recommended that you apply security on every piece of information that will be important to you and others; that applies in this case, too. Even when you are storing something seemingly unimportant, it may still contain data that will be useful to someone. It's always better to apply security and create strong passwords. Elasticsearch provides a plugin called `X-pack`, which eases the process of applying security for both the Elasticsearch engine and the Kibana tool.

Securing Elasticsearch

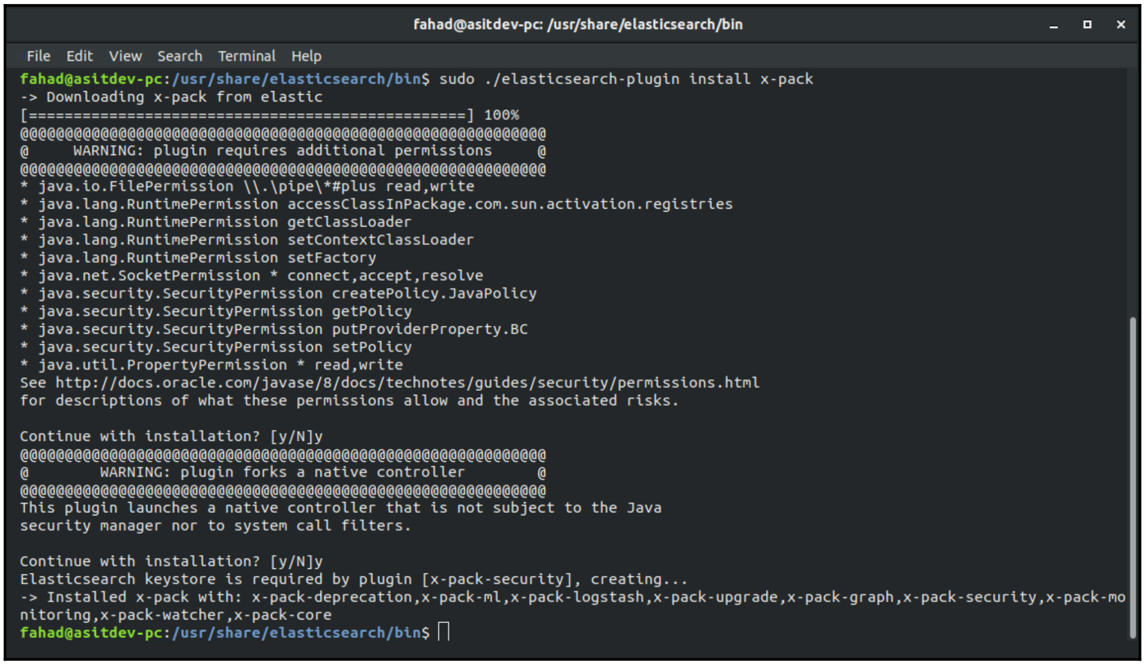
First of all, open up the terminal and execute the following command to browse the installation folder of Elasticsearch, which in our case is `/usr/share/elasticsearch/bin`:

```
$ cd /usr/share/elasticsearch/bin
```

The next step is to execute the following command to install the `x-pack` plugin for Elasticsearch:

```
$ sudo ./elasticsearch-plugin install x-pack
```

This will take some time to download and then install. Elasticsearch will display a message regarding permission; just press `Y` to proceed. The following screenshot displays the output generated by the preceding command:


fahad@asitdev-pc: /usr/share/elasticsearch/bin
File Edit View Search Terminal Help

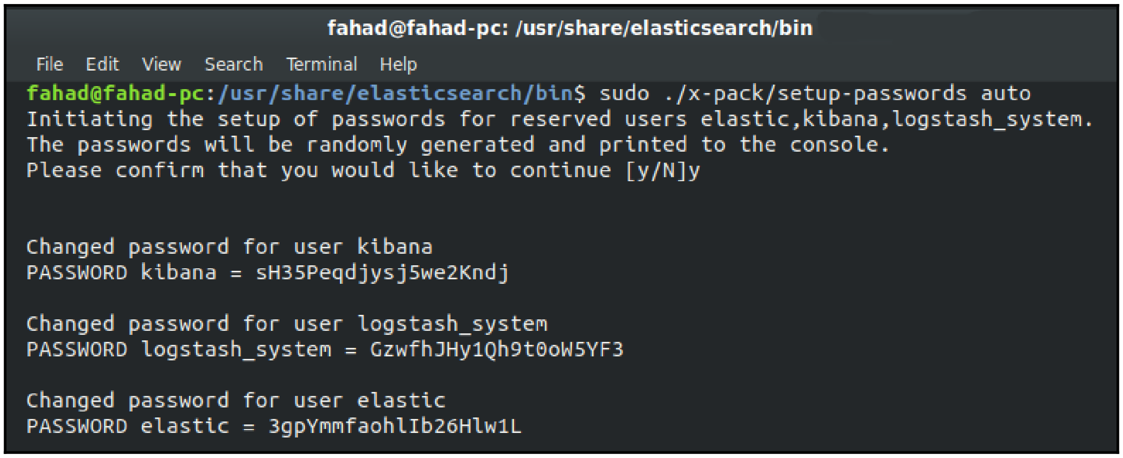
```
fahad@asitdev-pc:/usr/share/elasticsearch/bin$ sudo ./elasticsearch-plugin install x-pack
-> Downloading x-pack from elastic
[=================================================] 100%
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@     WARNING: plugin requires additional permissions     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* java.io.FilePermission \\.\pipe\*#plus read,write
* java.lang.RuntimePermission accessClassInPackage.com.sun.activation.registries
* java.lang.RuntimePermission getClassLoader
* java.lang.RuntimePermission setContextClassLoader
* java.lang.RuntimePermission setFactory
* java.net.SocketPermission * connect,accept,resolve
* java.security.SecurityPermission createPolicy.JavaPolicy
* java.security.SecurityPermission getPolicy
* java.security.SecurityPermission putProviderProperty.BC
* java.security.SecurityPermission setPolicy
* java.util.PropertyPermission * read,write
See http://docs.oracle.com/javase/8/docs/technotes/guides/security/permissions.html
for descriptions of what these permissions allow and the associated risks.

Continue with installation? [y/N]y
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        WARNING: plugin forks a native controller        @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
This plugin launches a native controller that is not subject to the Java
security manager nor to system call filters.

Continue with installation? [y/N]y
Elasticsearch keystore is required by plugin [x-pack-security], creating...
-> Installed x-pack with: x-pack-deprecation,x-pack-ml,x-pack-logstash,x-pack-upgrade,x-pack-graph,x-pack-security,x-pack-mo
nitoring,x-pack-watcher,x-pack-core
fahad@asitdev-pc:/usr/share/elasticsearch/bin$ 
```

X-Pack installation for Elasticsearch

Once it's finished, it's time to set up a password for Elasticsearch. Execute the following command to start the process:

```
$ sudo ./x-pack/setup-passwords auto
```

Once you have executed this command, it will generate a secure random password and set the security configuration accordingly. After finishing setup, it will print the randomly generated password on the terminal for your safe-keeping and any future usage, as shown in the following screenshot:


fahad@fahad-pc: /usr/share/elasticsearch/bin
File Edit View Search Terminal Help

```
fahad@fahad-pc:/usr/share/elasticsearch/bin$ sudo ./x-pack/setup-passwords auto
Initiating the setup of passwords for reserved users elastic,kibana,logstash_system.
The passwords will be randomly generated and printed to the console.
Please confirm that you would like to continue [y/N]y


Changed password for user kibana
PASSWORD kibana = sH35Peqdjysj5we2Kndj

Changed password for user logstash_system
PASSWORD logstash_system = GzwfhJHy1Qh9t0oW5YF3

Changed password for user elastic
PASSWORD elastic = 3gpYmmfaohlIb26Hlw1L
```

Elasticsearch security setup using the X-Pack plugin

Securing Kibana

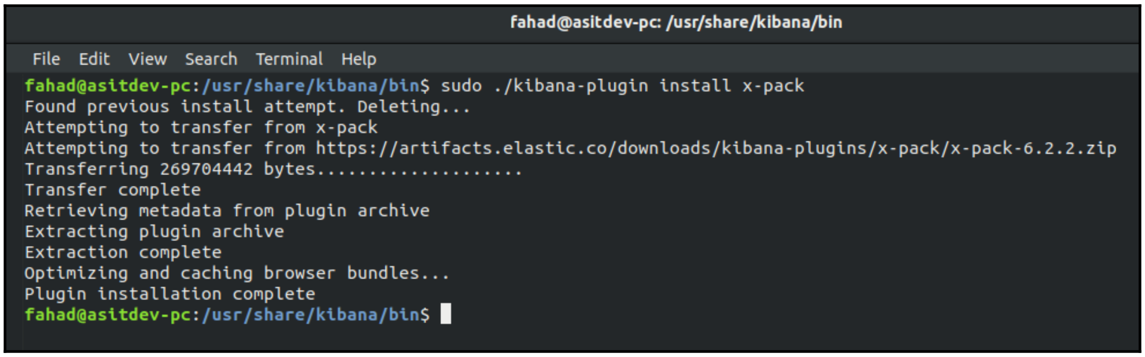
We will now install X-Pack for Kibana. Open up the terminal and execute the following command to browse the installation folder of Kibana, which in our case is `/usr/share/kibana/bin`:

```
$ cd /usr/share/kibana/bin
```

The next step is to execute the following command to install the X-Pack plugin for Kibana:

```
$ sudo ./kibana-plugin install x-pack
```

Similar to Elasticsearch, this will take some time to download and install. The following screenshot displays the output generated by the preceding command:


fahad@asitdev-pc: /usr/share/kibana/bin
File Edit View Search Terminal Help

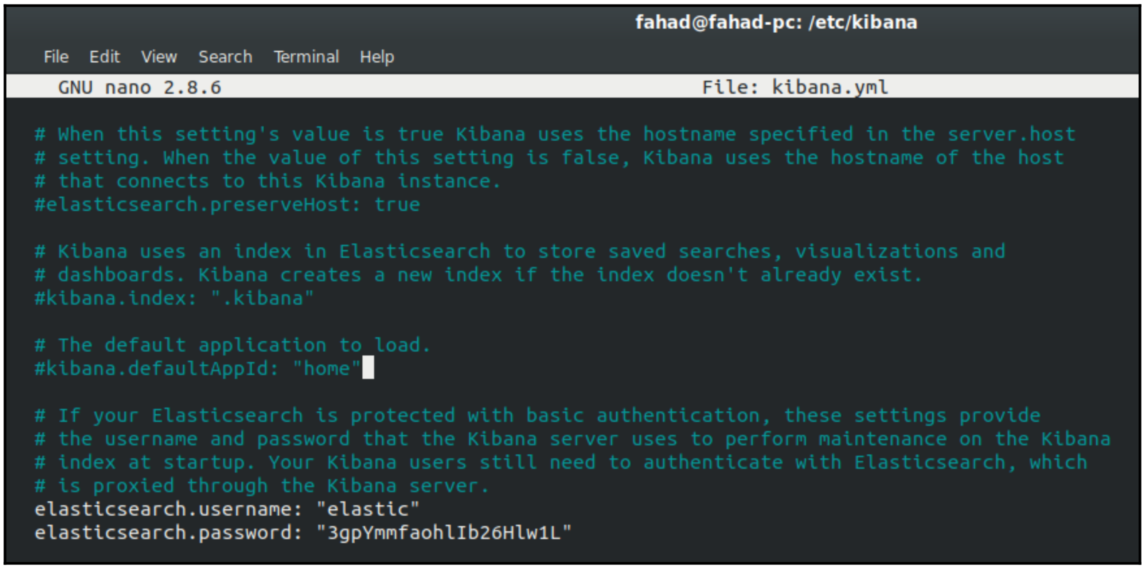
```
fahad@asitdev-pc:/usr/share/kibana/bin$ sudo ./kibana-plugin install x-pack
Found previous install attempt. Deleting...
Attempting to transfer from x-pack
Attempting to transfer from https://artifacts.elastic.co/downloads/kibana-plugins/x-pack/x-pack-6.2.2.zip
Transferring 269704442 bytes....................
Transfer complete
Retrieving metadata from plugin archive
Extracting plugin archive
Extraction complete
Optimizing and caching browser bundles...
Plugin installation complete
fahad@asitdev-pc:/usr/share/kibana/bin$
```

X-Pack installation for Kibana

Now that we have successfully installed X-Pack for Kibana, it's now time to configure it. Navigate to the directory `/etc/kibana`, where you will find the `kibana.yml` file. We need to update the newly generated Elasticsearch username and password in this configuration file so that they are validated with Elasticsearch the next time you start the Kibana web interface. Execute the following command to start editing the configuration file:

```
$ sudo nano kibana.yml
```

Once you have executed this command, it will open up the configuration file in editing mode. Look for the lines `elasticsearch.username` and `elasticsearch.password`. By default, both lines will be in a commented mode; just un-comment them by removing the first character, `#`, and defining the username and password we generated earlier, as shown in the following screenshot:


fahad@fahad-pc: /etc/kibana
File Edit View Search Terminal Help
GNU nano 2.8.6 File: kibana.yml

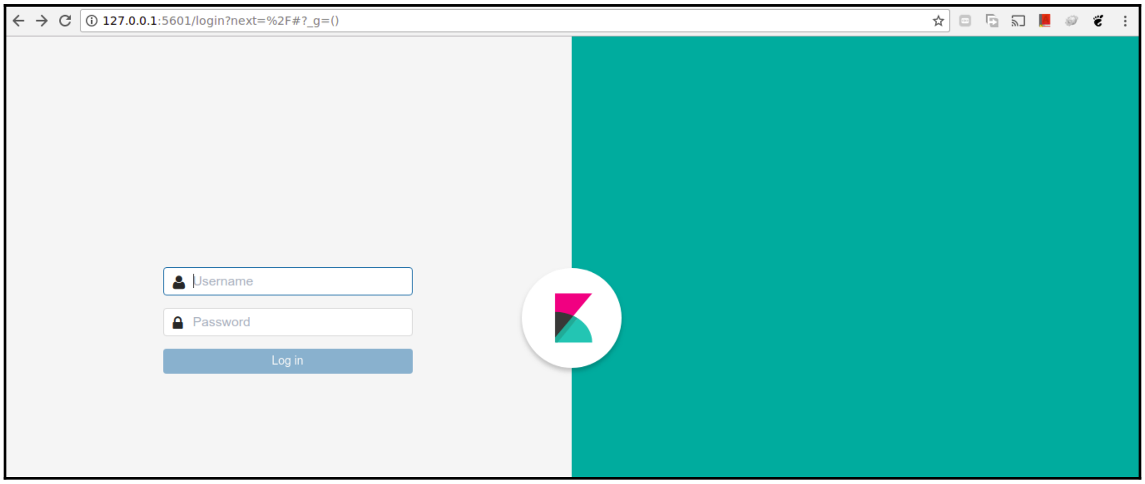
```
# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
#kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
elasticsearch.username: "elastic"
elasticsearch.password: "3gpYmmfaohlIb26Hlw1L"
```

Setting up the username and password for Elasticsearch in the Kibana configuration file

Now press *Ctrl* + *O* to write data and *Ctrl* + *X* to exit the editor. Now execute the following command to restart the Kibana service and to apply all of the necessary configurations:

```
$ sudo service kibana restart
```

Once the service is restarted, point your browser to `http://localhost:5601` to open Kibana. You should see a login page similar to the one shown in the following screenshot:



Kibana login page

Enter the username as `elastic` and the password as `3gpYmmfaohlIb26Hlw1L`, as generated earlier, to log in.

Now that we're all set up and ready, let's start using Elasticsearch to perform some CRUD operations.

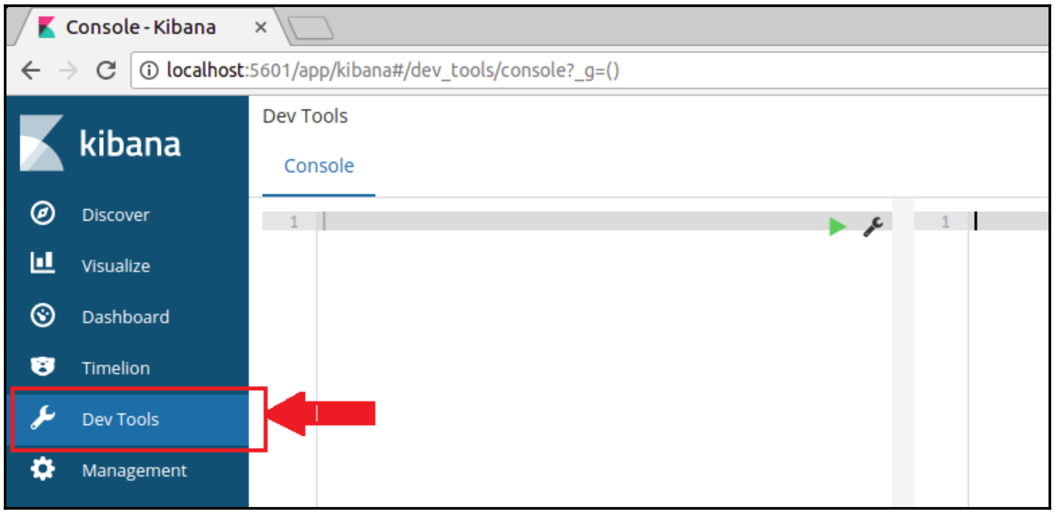
Understanding queries – CRUD commands

We already know that Elasticsearch uses a RESTful API as a communication model and we will be using Kibana as an IDE to run our API calls. Let's get started.

First, we need to start the Kibana service by executing the following commands, if Kibana has not already started:

```
$ sudo service kibana start
$ sudo service kibana status
```

Now point your browser to `http://localhost:5601/` to open the Kibana application, as we did earlier. Once the application is open, click on the **Dev Tools** options from the menu, as shown in the following screenshot:



Kibana Dev Tools

Before we execute commands, let's understand some of Kibana's basic concepts. Indexes are like tables in a traditional relational database but, as in other NoSQL databases, Elasticsearch is also schema-free. In the earlier versions of Elasticsearch, we had to create types along with every index. Now, starting from Version 6.0, you only have to specify a single type that will also be removed later on. For example, in earlier Versions, we created our indexes and types like this: `/student/contact`, `/student/courses`, `/student/grades`, and so on. From version 6.0 and later, indexes are created as follows: `/student-contact/detail`, `/student-grades/detail`, `/student-courses/details`, and so on.

In the chapter on backend architecture in this book, we studied RESTful API and its operations; let's revisit that. RESTful APIs have four operations, which are as follows:

- GET
- PUT
- POST
- DELETE

We will be using these operations to interact with Elasticsearch using Kibana. We should also note that we will be passing a JSON document along with the REST command. JSON supports six datatypes, which are as follows:

- String
- Number (this can be any number, an integer, a float, or a double)
- null
- Array, of any JSON-supported data type
- Boolean
- Another JSON document

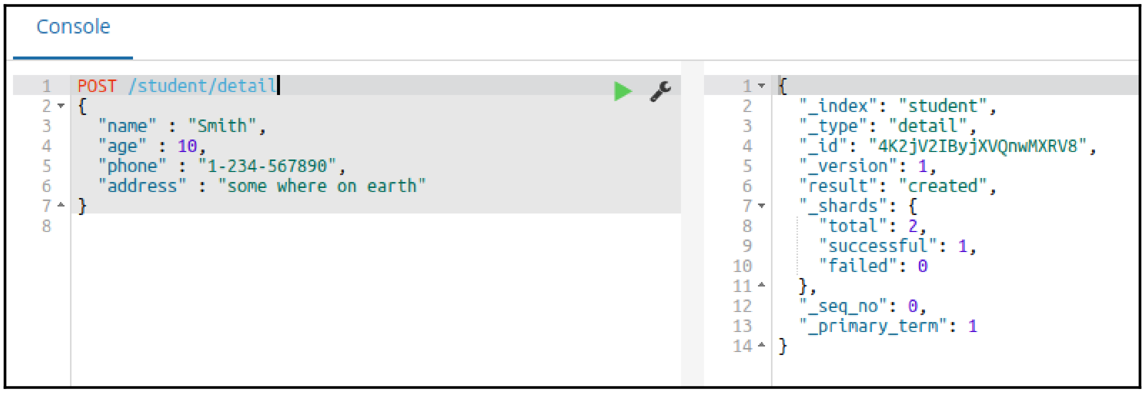
In the upcoming sections, we will follow a student database example. We will create documents for each student and their related information. Let's move on to executing queries and creating our first index in Elasticsearch.

Creating

(Note that the left-hand side box is for writing a query and the right-hand side box is where the output will appear.) Let's now execute our first query in order to create a student record. The JSON document in the query is self-explanatory. Execute the following code in Kibana:

```
POST /student/detail
{
  "name" : "Smith",
  "age" : 10,
  "phone" : "1-234-567890",
  "address" : "some where on earth"
}
```

The preceding code will create an index named `student` with the type `detail`, as shown in the following screenshot:



Console

```
POST /student/detail
{
  "name" : "Smith",
  "age" : 10,
  "phone" : "1-234-567890",
  "address" : "some where on earth"
}
```

```
{
  "_index": "student",
  "_type": "detail",
  "_id": "4K2jV2IByjXVQnwMXRV8",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

Query execution and its output in Elasticsearch using Kibana

If you look closely at the output, you will see the following information:

- `_index`: This contains the name of the index for which the related query was executed
- `_type`: This contains the name of the type for which the related query was executed
- `_id`: This is the ID of the record, which uniquely identifies an inserted document; if it is not provided in the query, one will be automatically generated
- `_version`: For every new record, the version will always be `1` – this will continue to increase when the same document is updated
- `_result`: This displays the action performed based on the provided query, such as created, deleted, or updated; in our case, it is `created`
- `_shards`: This contains the result of the query – whether it was successfully executed or not; if successful, the `_successful` flag will display `1` and the `_failure` flag `0` or vice versa

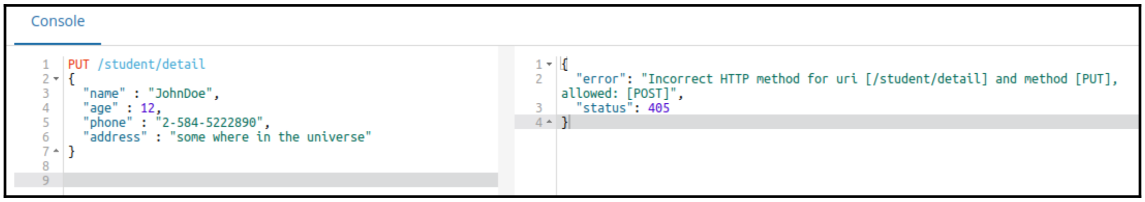
In this example, we have looked at only one way of adding a document in Elasticsearch, but there is another – we can use the `PUT` command. If you check the previous command, you'll see that we used POST. Why are there two different commands with the same purpose? If you use POST, Elasticsearch will automatically generate a unique ID for the inserted document, while if you use the `PUT` command, the executioner has to provide a unique ID in the command itself, the format for which is as follows:

```
PUT /student/detail/1
{
  "name" : "JohnDoe",
  "age" : 12,
  "phone" : "2-584-5222890",
  "address" : "some where in the universe"
}
```

Now, if you look closely at the preceding command's first line, you will see that after `/student/detail`, there is `/1`. This will become the unique ID for this document. The output of this command is as follows:

```
{
  "_index": "student",
  "_type": "detail",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 2,
  "_primary_term": 1
}
```

If you try to use the `PUT` command without providing a document ID, it will issue an error similar to the one as follows:



```
PUT /student/detail
{
  "name" : "JohnDoe",
  "age" : 12,
  "phone" : "2-584-5222890",
  "address" : "some where in the universe"
}
```

```
{
  "error": "Incorrect HTTP method for uri [/student/detail] and method [PUT], allowed: [POST]",
  "status": 405
}
```

The preceding error message tells us that if we cannot provide the document's ID, then we need to use the `POST` command, which is allowed as the written pattern.

Reading

Now that we have successful inserted our data into Elasticsearch, let's retrieve it. The simplest way of retrieving data is by using the document ID. In the previous example, we used a document ID of `1` when inserting a document in the `student` index. Let's now try to retrieve it by executing the following query:

```
GET /student/detail/1
```

The preceding query will come back with the following response:

```
{
  "_index": "student",
  "_type": "detail",
  "_id": "1",
  "_version": 6,
  "found": true,
  "_source": {
    "name": "JohnDoe",
    "age": 12,
    "phone": "2-584-5222890",
    "address": "some where in the universe"
  }
}
```

Now, if you want to retrieve all of the documents in the `student` index, execute the following query:

```
GET /student/detail/_search
```

The preceding command will display the following output:

```
{
  "took": 273,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 1,
    "hits": [
      {
        "_index": "student",
        "_type": "detail",
        "_id": "1",
        "_score": 1,
        "_source": {
          "name": "JohnDoe",
          "age": 12,
          "phone": "2-584-5222890",
          "address": "some where in the universe"
        }
      },
      {
        "_index": "student",
        "_type": "detail",
        "_id": "4K2jV2IByjXVQnwMXRV8",
        "_score": 1,
        "_source": {
          "name": "Smith",
          "age": 10,
          "phone": "1-234-567890",
          "address": "some where on earth"
        }
      }
    ]
  }
}
```

Here, we only inserted two documents, so only saw two documents in the output – one with a manually-entered ID and the other with the automated ID. Some other things to note are as follows:

- `"took"`: This is the amount of time it took Elasticsearch to fetch the documents in milliseconds
- `"timed_out"`: This tells us whether the query fetched all the available records or if it timed out
- `"hits" -> "total"`: This tells us the number of documents found

Updating

The `update` command is very much similar to the one for creation; the only difference is that, in the `POST` document, there needs to be another document called `doc` that contains any fields that need to be updated. Execute the `update` command as follows:

```
POST /student/detail/1/_update
{
  "doc" : {
      "phone" : "1-012-9876543"
  }
}
```

The preceding command will update the phone number of a student with the ID `1`. The output of this command is as follows:

```
{
  "_index": "student",
  "_type": "detail",
  "_id": "1",
  "_version": 2,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 5,
  "_primary_term": 1
}
```

If you look closely at this output, you will see that there is an additional flag to the successful one of `1`. The output also provides a version number, `2`, indicating that this is the first time the document has been updated since its creation, where it shows a version number of `1`.

Deleting

Deleting a document is very simple; just execute the following command:

```
DELETE /student/detail/1
```

Elasticsearch will display the following output as a result:

```
{
  "_index": "student",
  "_type": "detail",
  "_id": "1",
  "_version": 2,
  "result": "deleted",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 3,
  "_primary_term": 1
}
```

In the output, we see that the operation was performed successfully.

Summary

In this chapter, we learned about an enterprise-grade, full text-based search engine called Elasticsearch, which we successfully installed and configured. We then moved on to the Elasticsearch plugin Kibana, which is used to visualize data from Elasticsearch and to execute REST API calls in order to manipulate data within Elasticsearch. We also installed and configured Kibana.

Security is very important these days, which is why, instead of leaving our server insecure or using the default security options, we discussed a plugin provided by Elasticsearch, called X-Pack. We installed and configured X-Pack for Elasticsearch and Kibana.

We then moved on to the different API structures that perform CRUD operations so that we can create, read, update, and delete documents within Elasticsearch. We practically performed each scenario and studied the output generated in return.

In the next chapter, we will learn about structured data and how to manipulate it with the help of different open source tools.

15
Structured Data

Up until a few years ago, companies were only storing data that was important to them and directly linked either with their business or customers. These types of data are mostly structural in form, which means that each of the data records in the dataset available will have similar attributes. In most cases, they are also linked together and have different types of relationship between them.

There are many commercial and open source relational databases available in the market that help us to store and manipulate structural data. Example of these databases are MySQL, Microsoft SQL, and PostgreSQL. Some of them are free to use for both individual or commercial use, but some of them require licenses. The license cost varies from provider to provider based on the features they provide and support. The number of users and IoT devices is increasing rapidly, resulting in a flow of data with much faster frequency. It is very difficult to scale these databases, as most of them require expensive hardware and it is also costly to maintain them.

In previous chapters, we have already learned about Hadoop and its infrastructure requirements. So, instead of managing a cluster of expensive hardware that also keeps on growing, it's better to migrate these relational databases into Hadoop, which runs on commodity hardware. Hadoop is specifically built to handle large datasets with frequently growing data and it is also very scalable.

In this chapter, we will go through a step-by-step guide to migrate our structured data from the MySQL database into HBase. We discussed HBase in `Chapter 3`, *Hadoop Ecosystem*, and also performed a comparison between SQL and NoSQL databases in `Chapter 4`, *NoSQL Database*. The main benefit of using HBase is that it runs on Hadoop cluster and Hadoop runs on commodity hardware. It is very easy to scale Hadoop clusters and add more and more resources to it. On the other hand, if you scale a MySQL database, it will be very costly, as you will require expensive hardware as well as a lot of resources to manage it.

Let's start with details of the structured data that we will be migrating from MySQL to HBase, and the different entities and tables it contains.

Data analysis

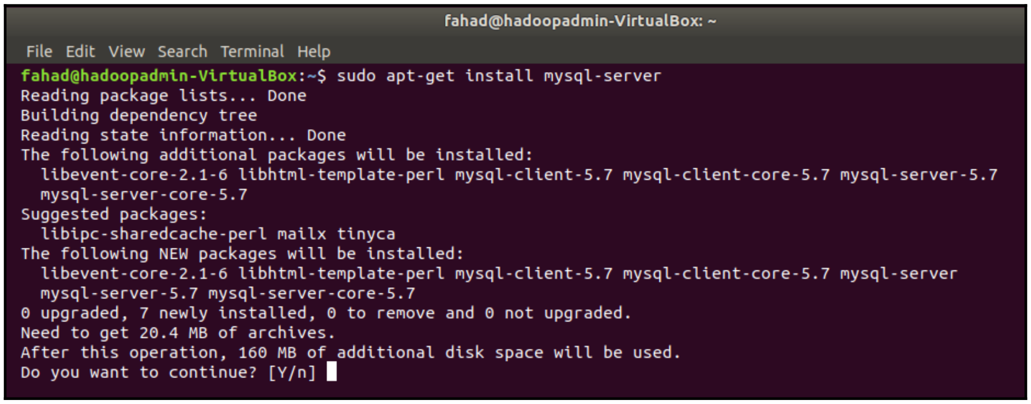
Let's now discuss the different tables available in our MySQL database that we will be exporting to our HBase database. We will be simulating an employee database, which will have information related to their personal information and the details of their salaries. In order to simulate a full scenario, we will also be installing MySQL database in our computer so that we can perform all of the actions from end to end in practical terms. So, let's start with the installation of a MySQL server.

Installing MySQL

Installing the MySQL database is pretty straightforward in Ubuntu. Just execute the following command to install it:

```
$ sudo apt-get install mysql-server
```

It will first prompt you to confirm to start the installation procedure, and then download approximately 20 MB of data, as shown in the following screenshot:

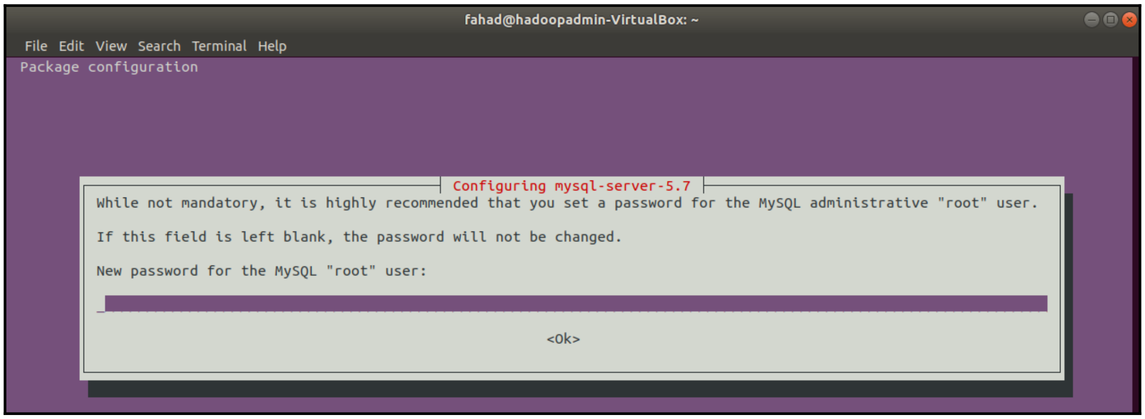


fahad@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

```
fahad@hadoopadmin-VirtualBox:~$ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7 mysql-client-core-5.7 mysql-server-5.7
  mysql-server-core-5.7
Suggested packages:
  libipc-sharedcache-perl mailx tinyca
The following NEW packages will be installed:
  libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7 mysql-client-core-5.7 mysql-server
  mysql-server-5.7 mysql-server-core-5.7
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 20.4 MB of archives.
After this operation, 160 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Starting MySQL installation

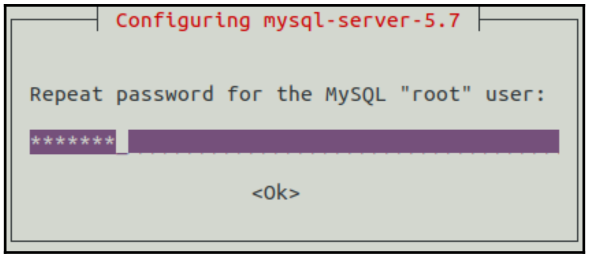
Once it downloads the files, it will prompt you for a password for the `root` user (which will have full administrative privileges over MySQL server), as shown in the following screenshot:



```
fahad@hadoopadmin-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
Package configuration

                    Configuring mysql-server-5.7
While not mandatory, it is highly recommended that you set a password for the MySQL administrative "root" user.

If this field is left blank, the password will not be changed.

New password for the MySQL "root" user:

_______________________________________

                              <Ok>
```

Setting the MySQL root user password

Once you enter the password, it will again prompt you to confirm the password, as shown in the following screenshot:



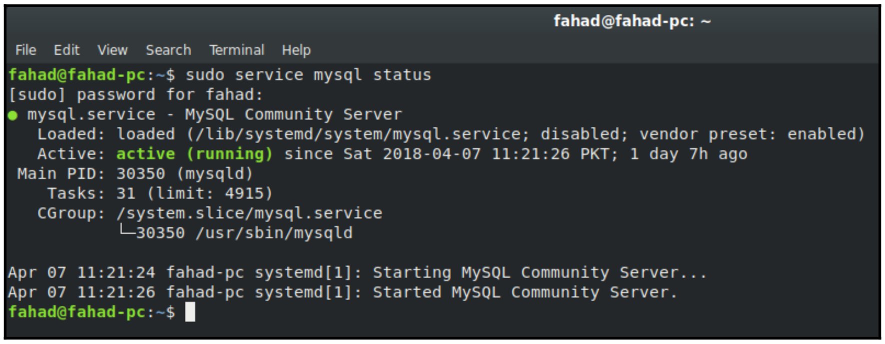
```
         Configuring mysql-server-5.7

Repeat password for the MySQL "root" user:

*******________________________________

                <Ok>
```

Confirming the root user password for MySQL Server

Once the installation completes, restart your computer. After restarting, execute the following command to verify MySQL server's installation `status`:

```
$ sudo service mysql status
```

It will display an output similar to the following screenshot:



```
fahad@fahad-pc:~$ sudo service mysql status
[sudo] password for fahad:
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-04-07 11:21:26 PKT; 1 day 7h ago
 Main PID: 30350 (mysqld)
    Tasks: 31 (limit: 4915)
   CGroup: /system.slice/mysql.service
           └─30350 /usr/sbin/mysqld

Apr 07 11:21:24 fahad-pc systemd[1]: Starting MySQL Community Server...
Apr 07 11:21:26 fahad-pc systemd[1]: Started MySQL Community Server.
fahad@fahad-pc:~$
```

MySQL Service Status

It is showing active (running), which means that the installation of MySQL was successful and is running in the background.

Importing data

Now that we have successfully installed MySQL and is running in the background, its now time to create our database and the related tables. For this purpose, copy the following code into a text file and save it as `employees.sql` in the `/usr/bin/` folder. It will require that we bulk load our MySQL server with the data that we will be exporting to our HBase database server later on:

```
-- Host: localhost Database: employees
-- ------------------------------------------------------
-- Server version 5.7.21-0ubuntu0.17.10.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
```

```
-- Current Database: `employees`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `employees` /*!40100 DEFAULT
CHARACTER SET latin1 */;

USE `employees`;

--
-- Table structure for table `empInfo`
--

DROP TABLE IF EXISTS `empInfo`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `empInfo` (
  `empId` int(11) NOT NULL,
  `empName` varchar(45) DEFAULT NULL,
  `phoneNo` varchar(45) DEFAULT NULL,
  `address` varchar(45) DEFAULT NULL,
  `designation` varchar(45) DEFAULT NULL,
  `department` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`empId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `empInfo`
--

LOCK TABLES `empInfo` WRITE;
/*!40000 ALTER TABLE `empInfo` DISABLE KEYS */;
INSERT INTO `empInfo` VALUES (1,'John Smith','123456789','somewhere on
earth','clerk','Accounts'),(2,'John Doe','987987999','somewhere on
universe','engineer','Development'),(3,'Smith','5432123456','somewhere','en
gineer','Development'),(4,'Andrew','331222345','under the
hood','manager','Development'),(5,'Troy','9102385731','nowhere','director',
'Business');
/*!40000 ALTER TABLE `empInfo` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `empSalary`
--

DROP TABLE IF EXISTS `empSalary`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `empSalary` (
  `salaryId` int(11) NOT NULL,
  `salaryMonth` varchar(10) DEFAULT NULL,
  `empId` int(11) DEFAULT NULL,
  `salaryAmount` double DEFAULT NULL,
  PRIMARY KEY (`salaryId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `empSalary`
--

LOCK TABLES `empSalary` WRITE;
/*!40000 ALTER TABLE `empSalary` DISABLE KEYS */;
INSERT INTO `empSalary` VALUES
(1,'2018-04',1,1025),(2,'2018-04',2,2556),(3,'2018-04',3,1500),(4,'2018-04'
,4,5000),(5,'2018-04',5,4780);
/*!40000 ALTER TABLE `empSalary` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Once you create this file, execute the following command to load it into MySQL:

```
$ mysql -u root -p < employees.sql
```

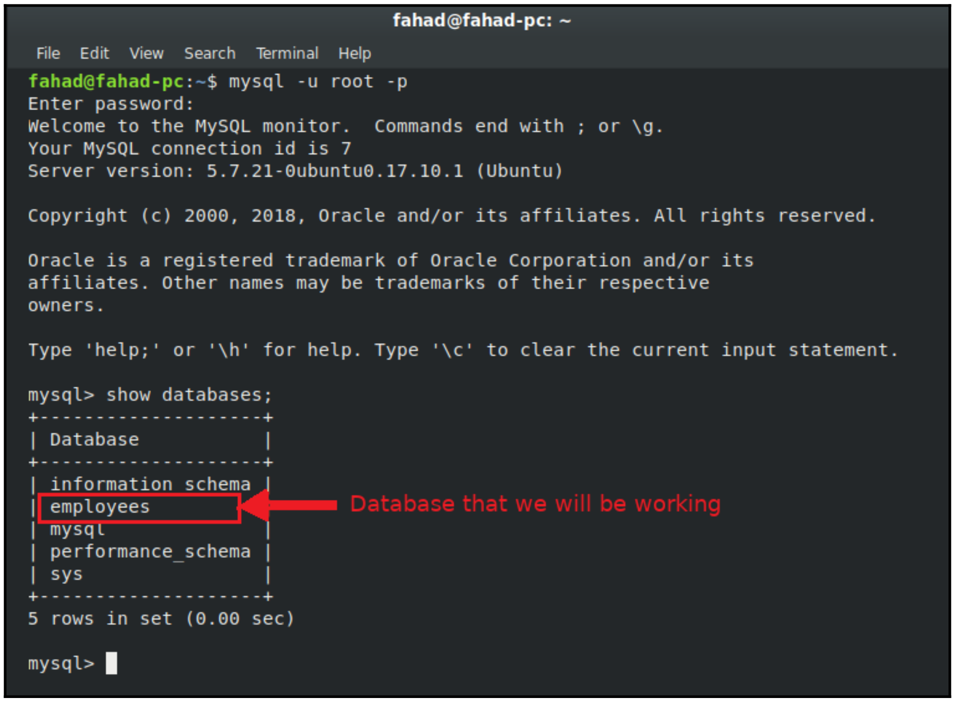
If it returns to prompt without any error, then it executed successfully and all the data is now available to be analyzed. Otherwise, make sure you haven't made any mistakes while creating the bulk load file and try to recreate it again.

Analyzing the data model

We will be using the MySQL terminal application to browse through different information. Execute the following command to log in to a MySQL instance:

```
$ mysql -u root -p
```

The -u parameter means using which user you want to log in, and the -p parameter means that this user has a password and you will provide it in the next step. The following screenshot shows the available databases in our MySQL instance, from which we will be using the employees database:


fahad@fahad-pc: ~
File Edit View Search Terminal Help
Database that we will be working

```
fahad@fahad-pc:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.21-0ubuntu0.17.10.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| employees          |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

Available databases in MySQL Instance

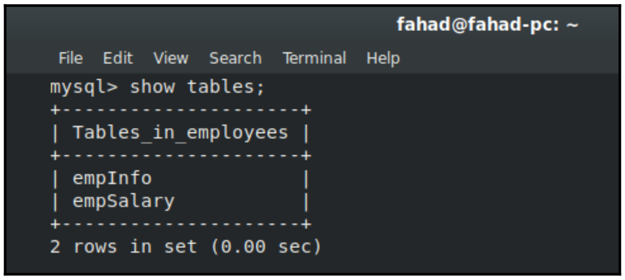
Now that we know we will be working with the employees database, execute the following command to select it as your working database:

```
mysql> use employees;
```

We will now check how many tables are available in this database. This is required, because we will be creating a MapReduce job against each table to map them between the MySQL table and the HBase table. Execute the following command to list all available tables in the current database:

```
mysql> show tables;
```

The following screenshot displays the available tables in the database:



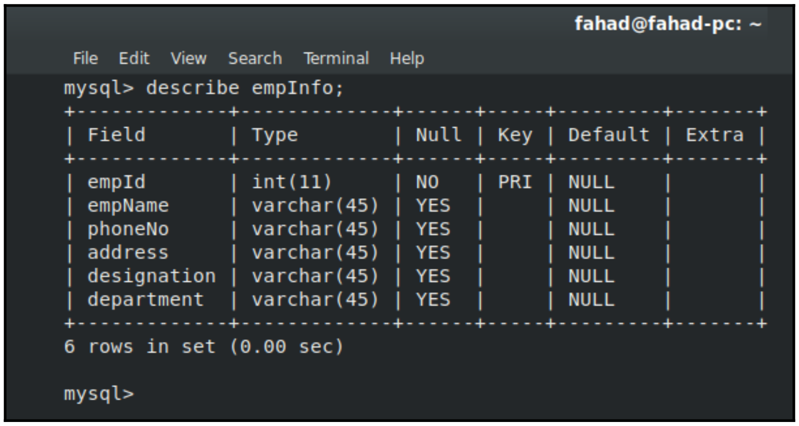
```
mysql> show tables;
+---------------------+
| Tables_in_employees |
+---------------------+
| empInfo             |
| empSalary           |
+---------------------+
2 rows in set (0.00 sec)
```

Available tables in the Employee Database

Now, we have the information that, in the current database, there are two tables; `empInfo` and `empSalary`. We will now be needing their column information, which will be required in a later stage, when we will create a program to copy the data from these tables. In order to get column details, execute the following command on the first table:

```
mysql> describe empInfo;
```

The following screenshot displays the information we need related to the `empInfo` table:



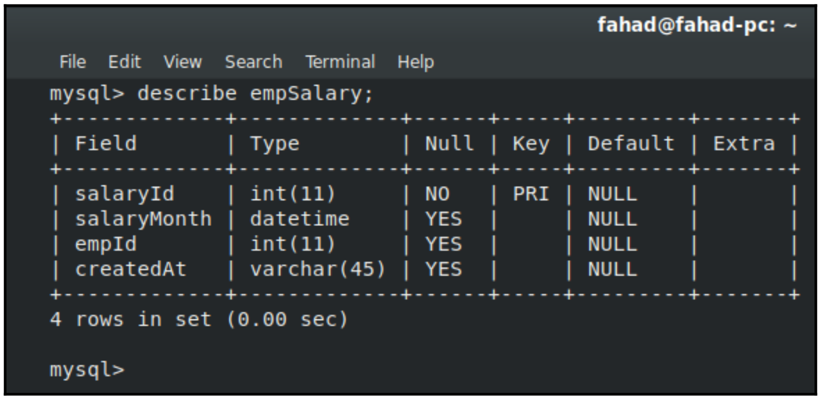
```
mysql> describe empInfo;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| empId       | int(11)     | NO   | PRI | NULL    |       |
| empName     | varchar(45) | YES  |     | NULL    |       |
| phoneNo     | varchar(45) | YES  |     | NULL    |       |
| address     | varchar(45) | YES  |     | NULL    |       |
| designation | varchar(45) | YES  |     | NULL    |       |
| department  | varchar(45) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql>
```

Column details of the empInfo table in the Employee Database

Similarly, now execute the following command to get the required details for the `empSalary` table:

```
mysql> describe empSalary;
```

The following screenshot displays the output of this command:



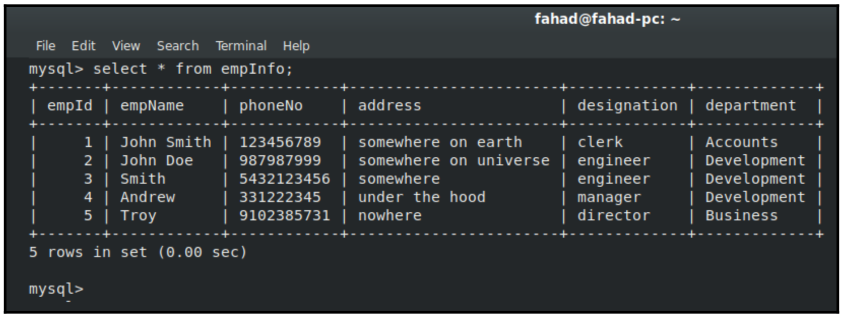
```
mysql> describe empSalary;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| salaryId    | int(11)     | NO   | PRI | NULL    |       |
| salaryMonth | datetime    | YES  |     | NULL    |       |
| empId       | int(11)     | YES  |     | NULL    |       |
| createdAt   | varchar(45) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Column details of the EmpSalary table

We will execute the following command to check whether we have any data available in the `empInfo` and `empSalary` tables one by one:

```
mysql> select * from empInfo;
```

The following screenshot displays the output of this command:



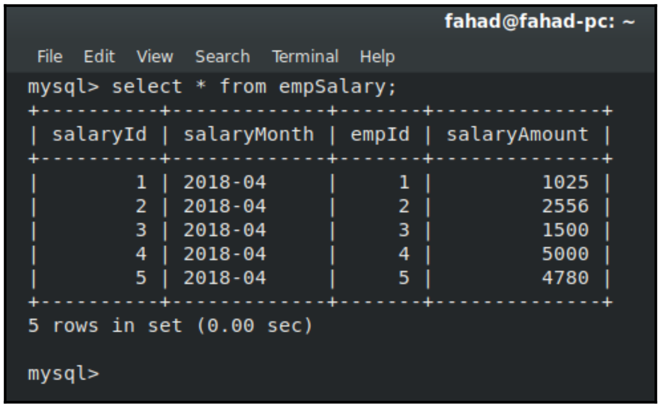
```
mysql> select * from empInfo;
+-------+------------+------------+----------------------+-------------+-------------+
| empId | empName    | phoneNo    | address              | designation | department  |
+-------+------------+------------+----------------------+-------------+-------------+
|     1 | John Smith | 123456789  | somewhere on earth   | clerk       | Accounts    |
|     2 | John Doe   | 987987999  | somewhere on universe| engineer    | Development |
|     3 | Smith      | 5432123456 | somewhere            | engineer    | Development |
|     4 | Andrew     | 331222345  | under the hood       | manager     | Development |
|     5 | Troy       | 9102385731 | nowhere              | director    | Business    |
+-------+------------+------------+----------------------+-------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

Data available in the empInfo table

I have put very little data in these tables so that it will be easier to understand and track the changes that we will be doing. Similarly, we will now execute the following command to check the available data in the `empSalary` table:

```
mysql> Select * from empSalary;
```

The following screenshot displays the output of this command:



```
mysql> select * from empSalary;
+----------+-------------+-------+--------------+
| salaryId | salaryMonth | empId | salaryAmount |
+----------+-------------+-------+--------------+
|        1 | 2018-04     |     1 |         1025 |
|        2 | 2018-04     |     2 |         2556 |
|        3 | 2018-04     |     3 |         1500 |
|        4 | 2018-04     |     4 |         5000 |
|        5 | 2018-04     |     5 |         4780 |
+----------+-------------+-------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Data available in the empSalary Table

Now that we have identified the structured and available data in our MySQL database, we will now prepare our HBase database to inject this data in it.

HBase

We learned about HBase in `Chapter 3`, *Hadoop Ecosystem*. We will now set up the HBase application in the Hadoop cluster that we already set up in `Chapter 2`, *Big Data Environment Setup*. If you have missed any of these, please revisit these chapters to get up to speed. We will now proceed with the installation of HBase.

Installation

In this first step, we will download the HBase application. At the time of writing, the current stable version of HBase is 1.4.3, and version 2.0 is in the beta 2 stage. We will be using the stable release to avoid encountering any bug or unknown failures. To learn more about different HBase releases, please visit `https://hbase.apache.org/`. We will now download HBase by executing the following command:

```
$ wget http://www-us.apache.org/dist/hbase/1.4.4/hbase-1.4.4-bin.tar.gz
```

This command will download HBase 1.4.4 in home directory. The complete path in our case is `/home/hadoopadmin/hbase-1.4.4-bin.tar.gz`. Now, execute the following command to extract the files in this archive:

```
$ tar -zxvf hbase-1.4.4-bin.tar.gz
```

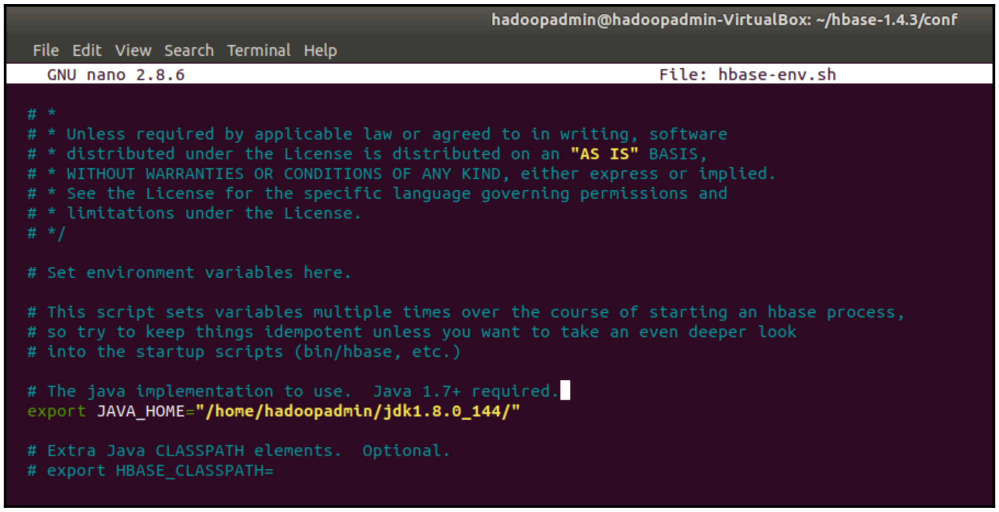
This will extract all the files to a folder named `hbase-1.4.3`. It's now time to do some HBase related configurations. The first file that we will be editing is the `hbase-env.sh` file, which is present in the `conf` folder in the HBase directory. Browse to this folder and execute the following command in order to start editing it:

```
$ cd ~/hbase-1.4.4/conf
$ sudo nano hbase-env.sh
```

Once you execute this command, it will allow you to edit the file. We need to set the `JAVA_HOME` variable to point to the correct installation directory. In our case, we will install Java in the `/home/hadoopadmin/jdk1.8.0_144/` directory. We will uncomment the line that contains the `JAVA_HOME` export variable and set it as follows:

```
export JAVA_HOME="/home/hadoopadmin/jdk1.8.0_144/"
```

The following screenshot shows the content of the file after setting the `JAVA_HOME` variable:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/conf
File Edit View Search Terminal Help
GNU nano 2.8.6
File: hbase-env.sh

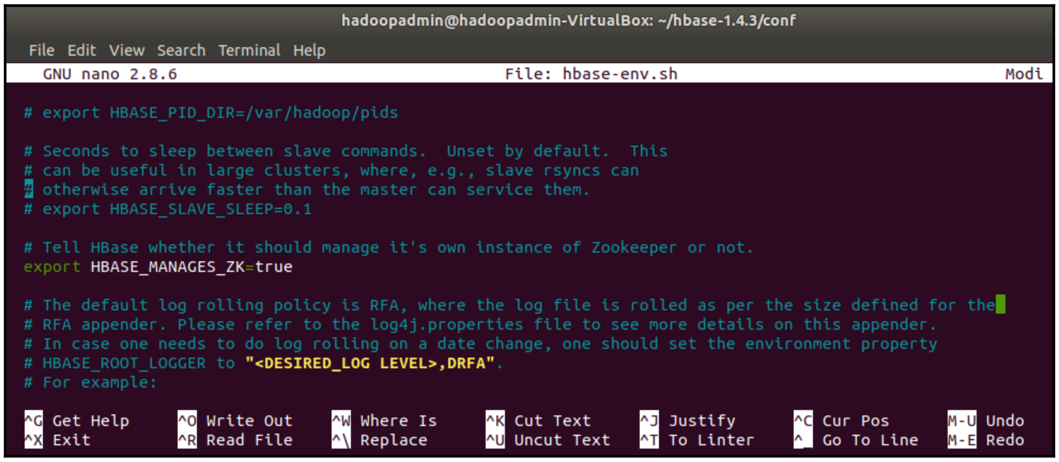
```
# *
# * Unless required by applicable law or agreed to in writing, software
# * distributed under the License is distributed on an "AS IS" BASIS,
# * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# * See the License for the specific language governing permissions and
# * limitations under the License.
# */

# Set environment variables here.

# This script sets variables multiple times over the course of starting an hbase process,
# so try to keep things idempotent unless you want to take an even deeper look
# into the startup scripts (bin/hbase, etc.)

# The java implementation to use.  Java 1.7+ required.
export JAVA_HOME="/home/hadoopadmin/jdk1.8.0_144/"

# Extra Java CLASSPATH elements.  Optional.
# export HBASE_CLASSPATH=
```

Setting up JAVA_HOME in HBase installation

Now, go to the end of this file, look for the following line, and uncomment it. This line enables HBase to manage its own Zookeeper instance:

```
export HBASE_MANAGES_ZK=true
```

The following screenshot illustrates this step:



```
hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/conf
File  Edit  View  Search  Terminal  Help
  GNU nano 2.8.6                          File: hbase-env.sh                                   Modi

# export HBASE_PID_DIR=/var/hadoop/pids

# Seconds to sleep between slave commands.  Unset by default.  This
# can be useful in large clusters, where, e.g., slave rsyncs can
# otherwise arrive faster than the master can service them.
# export HBASE_SLAVE_SLEEP=0.1

# Tell HBase whether it should manage it's own instance of Zookeeper or not.
export HBASE_MANAGES_ZK=true

# The default log rolling policy is RFA, where the log file is rolled as per the size defined for the
# RFA appender. Please refer to the log4j.properties file to see more details on this appender.
# In case one needs to do log rolling on a date change, one should set the environment property
# HBASE_ROOT_LOGGER to "<DESIRED_LOG LEVEL>,DRFA".
# For example:

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter   ^_ Go To Line  M-E Redo
```

Enabling HBase to manage its own Zookeeper instance

Now press `Ctrl+O` to write the file and `Ctrl+X` to exit the editor. Now, the next step is to create the directory that HBase will use to store its data files. We will create a new directory named `datafiles` inside the HBase directory by executing the following command:

```
$ cd ~/hbase-1.4.4/
$ mkdir datafiles
```

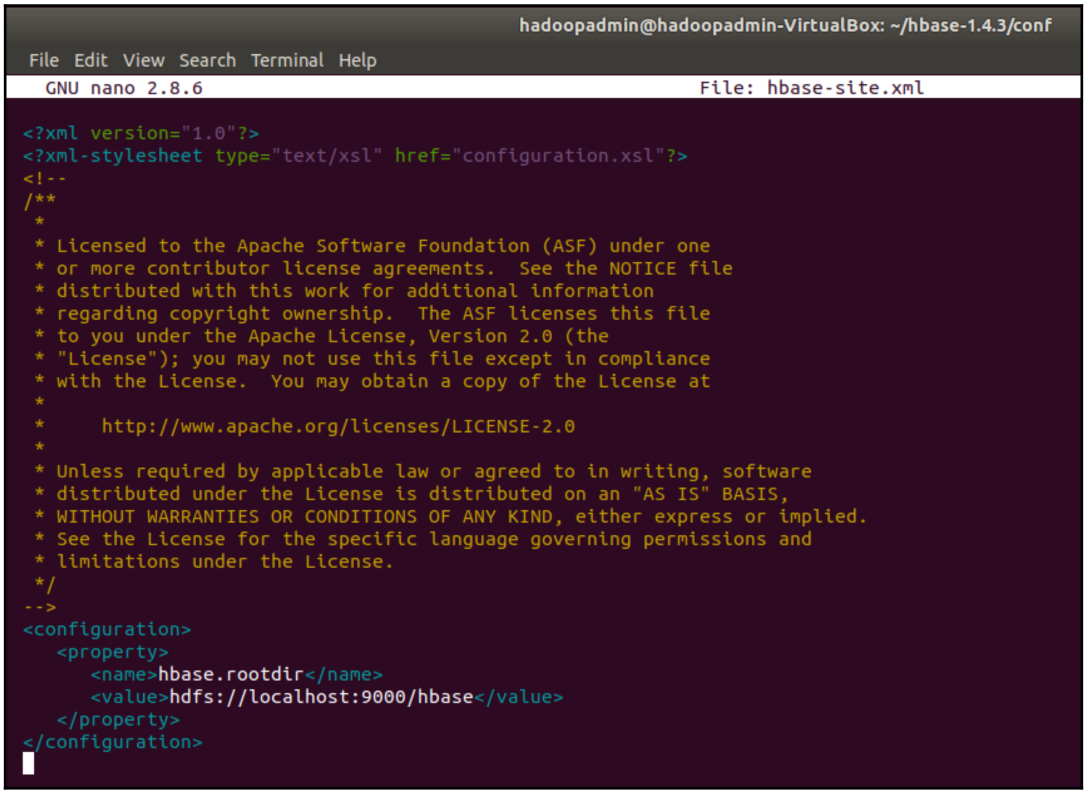
We will now edit the `hbase-site.xml` file to define the `hbase.rootdir` property so that it will start storing all the data files in it. Execute the following command to edit the file:

```
$ cd ~/hbase-1.4.4/conf
$ sudo nano hbase-site.xml
```

This will open up the editor to let you start editing. Copy the following code in between the `<configuration></configuration>` tags:

```
<property>
      <name>hbase.rootdir</name>
      <value>hdfs://localhost:9000/hbase</value>
</property>
```

The following screenshot shows the actual content of the `hbase-site.xml` file:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/conf
File Edit View Search Terminal Help
GNU nano 2.8.6
File: hbase-site.xml

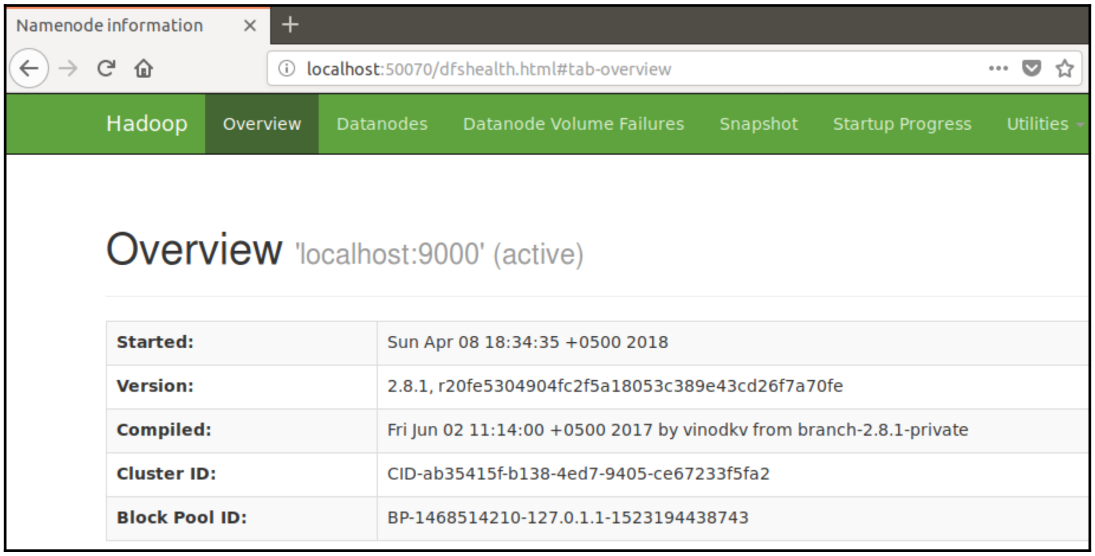
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
/**
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
-->
<configuration>
   <property>
      <name>hbase.rootdir</name>
      <value>hdfs://localhost:9000/hbase</value>
   </property>
</configuration>
```

Content of hbase-site.xml to set the rootdir property

We have finished all the related configuration for HBase and we are now ready to start the server.

Starting an HBase instance

Now that we have completed the configuration of HBase, it's time to start it. Before doing so, make sure that the Hadoop instance is up and running. To verify it, point your browser to `http://localhost:50070/`, which will display a page similar to the following screenshot:



Overview 'localhost:9000' (active)

Started:	Sun Apr 08 18:34:35 +0500 2018
Version:	2.8.1, r20fe5304904fc2f5a18053c389e43cd26f7a70fe
Compiled:	Fri Jun 02 11:14:00 +0500 2017 by vinodkv from branch-2.8.1-private
Cluster ID:	CID-ab35415f-b138-4ed7-9405-ce67233f5fa2
Block Pool ID:	BP-1468514210-127.0.1.1-1523194438743

Hadoop Server Overview

If you are able to view an output similar to this, it means that Hadoop is working fine and you can proceed to start HBase. If it is giving any error messages, please go to the Hadoop installation guide provided in `Chapter 2`, *Big Data Environment Setup*. Once resolved, please proceed to the following instructions.

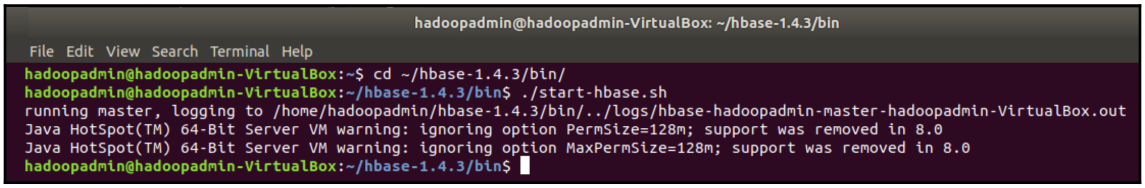
Make sure that you have created the `hbase` directory in your Hadoop file system, because as per the configuration done earlier, we have defined that HBase will store its data files in HDFS. Execute the following command in order to create it, if this has not already been done:

```
$ hdfs dfs -mkdir /hbase
```

We will now be executing the commands that are available in HBase's installation directory, so make sure that you are in the HBase directory. Execute the following command to start HBase:

```
$ cd ~/hbase-1.4.4/bin
$ ./start-hbase.sh
```

Once you execute this command, it will give you an output similar to the following screenshot:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/bin
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~$ cd ~/hbase-1.4.3/bin/
hadoopadmin@hadoopadmin-VirtualBox:~/hbase-1.4.3/bin$ ./start-hbase.sh
running master, logging to /home/hadoopadmin/hbase-1.4.3/bin/../logs/hbase-hadoopadmin-master-hadoopadmin-VirtualBox.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
hadoopadmin@hadoopadmin-VirtualBox:~/hbase-1.4.3/bin$ 
```

Start HBase

As you can see in the output `running master`, you have successfully started your HBase instance.

Stopping a HBase instance

In order to stop HBase, execute the following command:

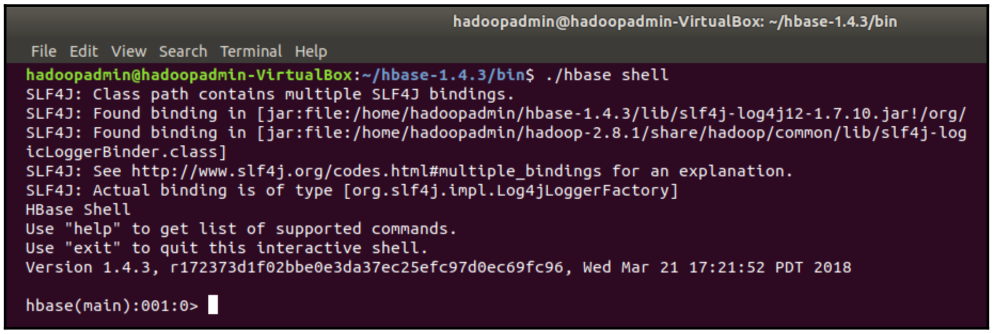
```
$ cd ~/hbase-1.4.4/bin
$ ./stop-hbase.sh
```

Preparing an HBase for migration

We will now start the HBase interactive shell to start our setup for importing data from MySQL. Execute the following command to open the HBase shell:

```
$ cd ~/hbase-1.4.4/bin
$ ./hbase shell
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/bin
File Edit View Search Terminal Help

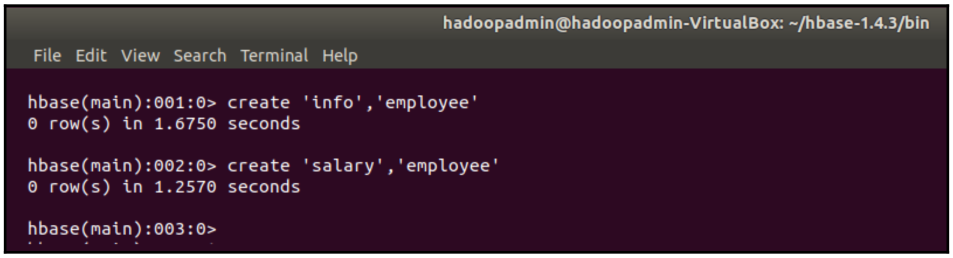
```
hadoopadmin@hadoopadmin-VirtualBox:~/hbase-1.4.3/bin$ ./hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hbase-1.4.3/lib/slf4j-log4j12-1.7.10.jar!/org/
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hadoop-2.8.1/share/hadoop/common/lib/slf4j-log
icLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.3, r172373d1f02bbe0e3da37ec25efc97d0ec69fc96, Wed Mar 21 17:21:52 PDT 2018

hbase(main):001:0>
```

Starting HBase interactive shell

The data model that we have in MySQL contains two tables, `empInfo` and `empSalary`. We will now need to create these two tables in HBase so that we can then copy the data from the MySQL tables to the HBase tables. Execute the following command to create the tables in HBase:

```
hbase(main):001:0> create 'info','employee'
hbase(main):002:0> create 'salary','employee'
```

The following screenshot displays the output we get after executing this command:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/bin
File Edit View Search Terminal Help

```
hbase(main):001:0> create 'info','employee'
0 row(s) in 1.6750 seconds

hbase(main):002:0> create 'salary','employee'
0 row(s) in 1.2570 seconds

hbase(main):003:0>
```

Creating tables in HBase

We have now successfully prepared our HBase database and are now ready to move on to the next step. Now, there are two possible scenarios; we can either write a MapReduce Java program, or we can use an open source application named Sqoop to copy data from the MySQL database to HBase. Here, we will be using Sqoop.

Sqoop

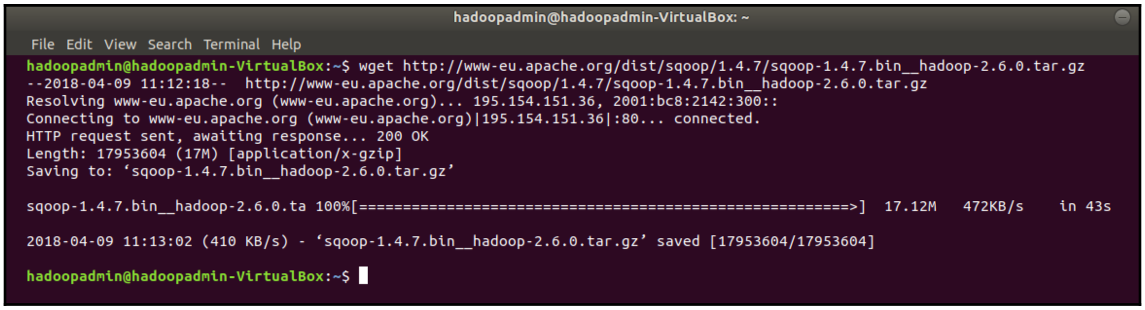
Sqoop is an open source tool that is used to import and export data between different relational databases and Hadoop ecosystems. As you know, the stable version of Sqoop is 1.4.7 and we will be using this version in our scenario. It is managed under the umbrella of Apache Projects. Let's start with the installation of Sqoop.

Installation

The installation of Sqoop is pretty straightforward. First, download the Sqoop application by executing the following command:

```
$ cd ~/
$ wget
http://www.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
```

The following screenshot illustrates the output of these commands:


hadoopadmin@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~$ wget http://www-eu.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
--2018-04-09 11:12:18--  http://www-eu.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
Resolving www-eu.apache.org (www-eu.apache.org)... 195.154.151.36, 2001:bc8:2142:300::
Connecting to www-eu.apache.org (www-eu.apache.org)|195.154.151.36|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17953604 (17M) [application/x-gzip]
Saving to: ‘sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz’

sqoop-1.4.7.bin__hadoop-2.6.0.ta 100%[======================================================>]  17.12M   472KB/s    in 43s

2018-04-09 11:13:02 (410 KB/s) - ‘sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz’ saved [17953604/17953604]

hadoopadmin@hadoopadmin-VirtualBox:~$
```

Download the Sqoop application

Now, execute the following command to extract the `tar` archive:

```
$ tar -zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
```

This will create a folder called `sqoop-1.4.7.bin__hadoop-2.6.0` and extract all the files to it. It's now time to do the related configuration. Execute the following command to edit the `bashrc` file in order to include the Sqoop installation folder in the system path.

```
$ sudo nano ~/.bashrc
```

Now, copy the following lines at the end of this file:

```
# Sqoop
export SQOOP_HOME="/home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0"
export PATH=$PATH:$SQOOP_HOME/bin
```

Now, press *Ctrl* + *O* to write the file and then *Ctrl* + *X* to exit from the editor. Then, execute the following to reload the `bashrc` file:

```
$ source ~/.bashrc
```

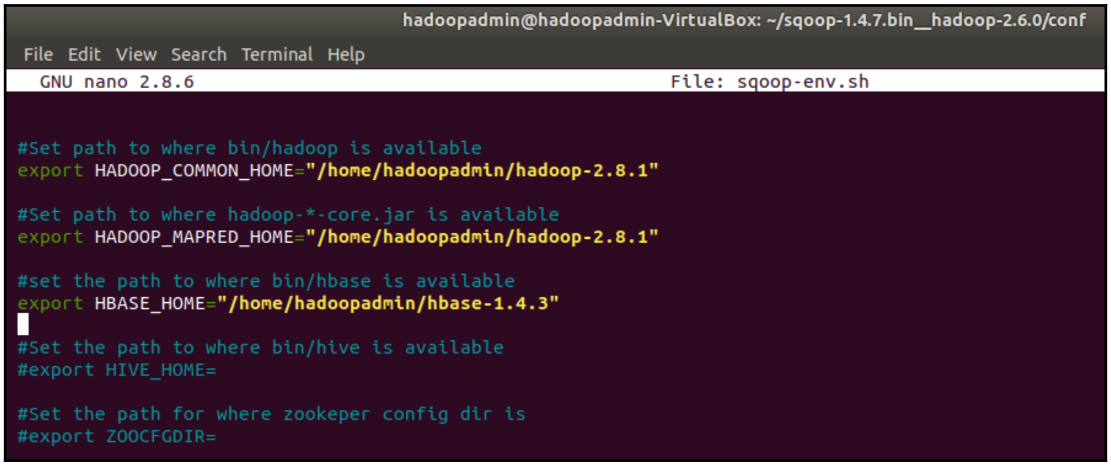
Now that we have reloaded the `bashrc` file, it is able to use the Sqoop variable in the system path. We will now define the Hadoop and HBase installation directory path in the Sqoop configuration file. Execute the following command to configure Sqoop:

```
$ cd $SQOOP_HOME/conf
$ cp sqoop-env-template.sh sqoop-env.sh
$ sudo nano sqoop-env.sh
```

This will open up the editor to set the following three variables:

- `HADOOP_COMMON_HOME`
- `HADOOP_MAPRED_HOME`
- `HBASE_HOME`

The following screenshot displays this configuration in our case:


hadoopadmin@hadoopadmin-VirtualBox: ~/sqoop-1.4.7.bin__hadoop-2.6.0/conf
File Edit View Search Terminal Help

```
  GNU nano 2.8.6                                   File: sqoop-env.sh


#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME="/home/hadoopadmin/hadoop-2.8.1"

#Set path to where hadoop-*-core.jar is available
export HADOOP_MAPRED_HOME="/home/hadoopadmin/hadoop-2.8.1"

#set the path to where bin/hbase is available
export HBASE_HOME="/home/hadoopadmin/hbase-1.4.3"

#Set the path to where bin/hive is available
#export HIVE_HOME=

#Set the path for where zookeper config dir is
#export ZOOCFGDIR=
```

Configure Hadoop and the HBase installation path in Sqoop env file

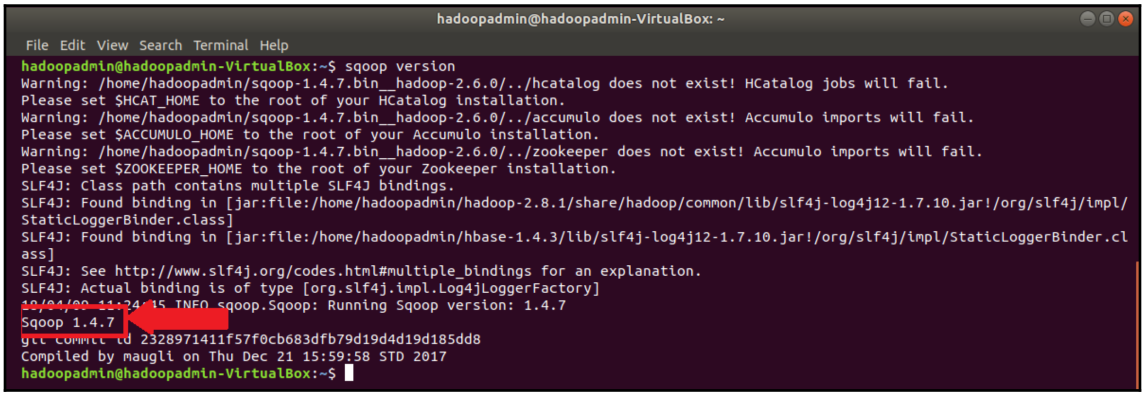
We have now completed the installation of Sqoop. It's now time to verify whether it has installed correctly.

Verifying the installation

Execute the following command to check the version of the current Sqoop installation:

```
$ sqoop version
```

The following screenshot displays the output of this command, mentioning the `version` number of the Sqoop that we have installed just now:


hadoopadmin@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~$ sqoop version
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hadoop-2.8.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hbase-1.4.3/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/04/09 11:24:45 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
Sqoop 1.4.7
git commit id 2328971411f57f0cb683dfb79d19d4d19d185dd8
Compiled by maugli on Thu Dec 21 15:59:58 STD 2017
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Checking the version number of the Sqoop installation

Now that we have successfully installed Sqoop and verified that everything is working as expected, it's time to start using it to import data from MySQL into HBase in Hadoop.

MySQL JDBC driver

We will be importing MySQL data in HBase. For that, we will require the MySQL JDBC driver class to be available in Sqoop's `lib` directory in order to be utilized. It is available officially from the MySQL website. Point your browser to `https://dev.mysql.com/downloads/connector/j/5.1.html` and download the **Platform Independent (Architecture Independent), Compressed TAR Archive**. It is nearly 4.6 MB in size. Once the download is finished, copy the file to your `Home` folder and execute the following command to extract it:

```
$  tar -zxvf mysql-connector-java-5.1.46
```

It will extract all the files to a folder named `mysql-connector-java-5.1.46`. Browse into this folder and execute the following command to copy the JAR file into Sqoop's `lib` folder:

```
$ cd mysql-connector-java-5.1.46/
$ cp mysql-connector-java-5.1.46-bin.jar $SQOOP_HOME/lib
```

We are now all set and ready to connect to our MySQL instance from Sqoop.

Importing data

Now, we have completed the setup and configuration required to import data from MySQL tables to HBase tables. First, we will create a folder. Sqoop will place all temporary files by executing the following command:

```
$ cd ~/
$ mkdir sqoopfiles
$ cd sqoopfiles
```

Let's now proceed to execute a `Sqoop` command to start the importing procedure:

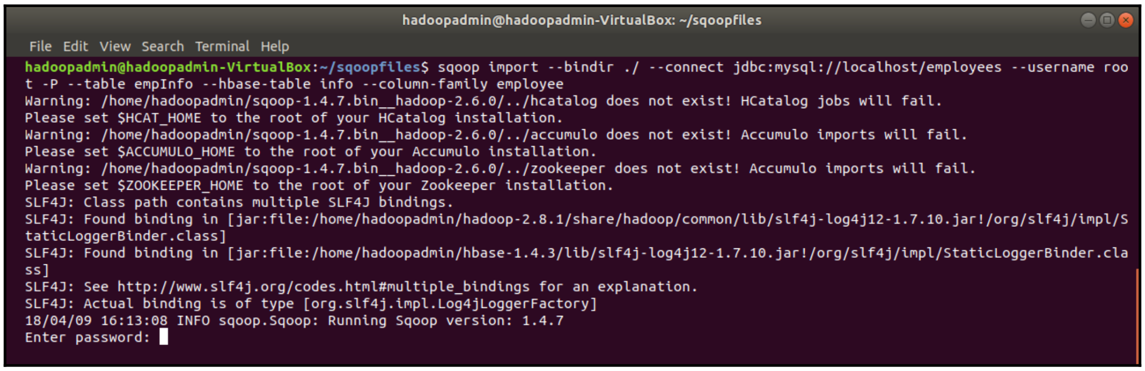
```
$ sqoop import
    --bindir ./
    --connect jdbc:mysql://localhost/employees
    --username root
    -P
    --table empInfo
    --hbase-table info
    --column-family employee
```

Let's first understand all the parameters that this command contains, as follows:

- `--bindir`: Specifies the current working directory for Sqoop. Based on the other parameters, sqoop will create a JAR file that contains the MapReduce job for importing data. Sqoop will place all the temporary files in it until the job is done.
- `--connect`: Defines the type of connection we are going to make and to which database. Sqoop supports many different relational databases, and with the proper connection drivers we can use Sqoop to import data from any database. In this example, we will be connecting to the MySQL `employees` database using the JDBC driver class.

- `--username`: Specifies the username that we will use to connect to the MySQL database.
- `-P`: Specifies that the MySQL connection will require a password to connect, which we will provide once the actual connection has been made.
- `--table`: The name of the MySQL table that we will be importing.
- `--hbase-table`: The name of the HBase table to which the data will be copied.
- `--column-family`: Specifies which column family the HBase table belongs to.

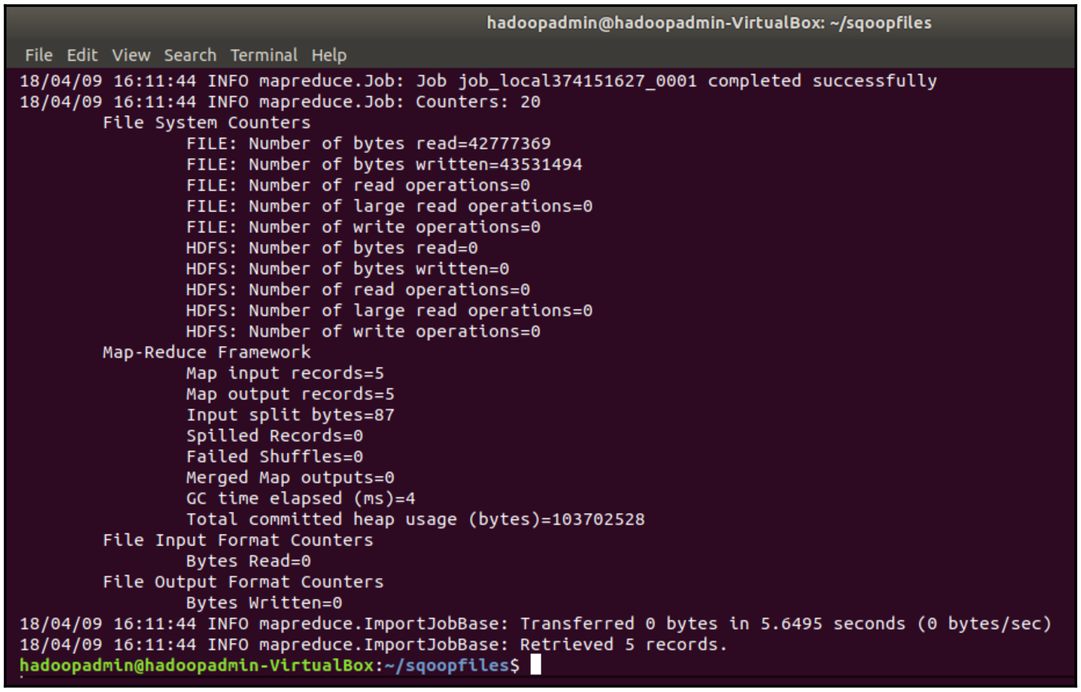
Now that we have understood all the parameters, let's execute this command. The following screenshot displays the prompt that asks for the MySQL password once we execute this command:



```
hadoopadmin@hadoopadmin-VirtualBox:~/sqoopfiles$ sqoop import --bindir ./ --connect jdbc:mysql://localhost/employees --username roo
t -P --table empInfo --hbase-table info --column-family employee
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/hadoopadmin/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hadoop-2.8.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/S
taticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hbase-1.4.3/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cla
ss]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/04/09 16:13:08 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
Enter password: 
```

Sqoop prompting to provide the MySQL Password to proceed

Once we provide the password, it will proceed to the next step and create different classes and JAR files, which you can view in the `sqoopfile` folder once the job finishes. The following screenshot displays the output when Sqoop finishes executing:


hadoopadmin@hadoopadmin-VirtualBox: ~/sqoopfiles
File Edit View Search Terminal Help

```
18/04/09 16:11:44 INFO mapreduce.Job: Job job_local374151627_0001 completed successfully
18/04/09 16:11:44 INFO mapreduce.Job: Counters: 20
        File System Counters
                FILE: Number of bytes read=42777369
                FILE: Number of bytes written=43531494
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=0
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=0
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=4
                Total committed heap usage (bytes)=103702528
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
18/04/09 16:11:44 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 5.6495 seconds (0 bytes/sec)
18/04/09 16:11:44 INFO mapreduce.ImportJobBase: Retrieved 5 records.
hadoopadmin@hadoopadmin-VirtualBox:~/sqoopfiles$
```

Sqoop finishes importing data from MySQL into HBase

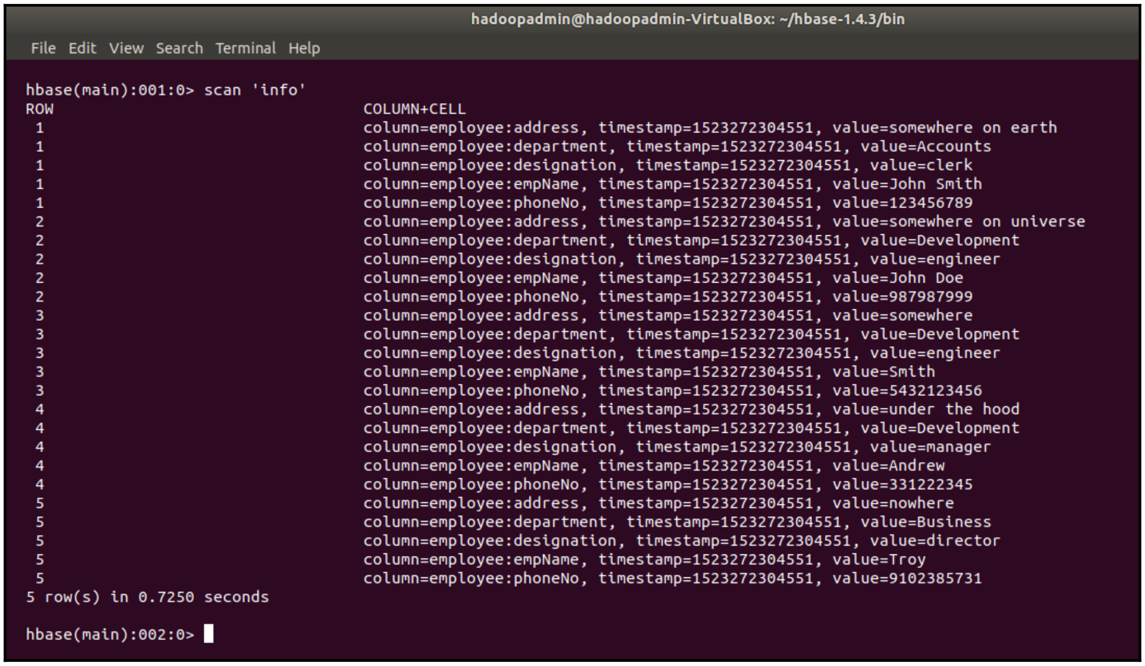
If you now look at the last line of the output, it says `Retrieved 5 records`. This means that Sqoop found five records in the MySQL `empInfo` table and successfully imported them into the HBase `info` table.

Verifying the imported data

As per Sqoop, it has successfully imported the data into HBase. It's now time to verify it from HBase itself. For this purpose, fire up the `hbase` shell as shown earlier in this chapter and execute the following command in it:

```
hbase(main):001:0> scan 'info'
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~/hbase-1.4.3/bin
File Edit View Search Terminal Help

```
hbase(main):001:0> scan 'info'
ROW                                COLUMN+CELL
 1                                 column=employee:address, timestamp=1523272304551, value=somewhere on earth
 1                                 column=employee:department, timestamp=1523272304551, value=Accounts
 1                                 column=employee:designation, timestamp=1523272304551, value=clerk
 1                                 column=employee:empName, timestamp=1523272304551, value=John Smith
 1                                 column=employee:phoneNo, timestamp=1523272304551, value=123456789
 2                                 column=employee:address, timestamp=1523272304551, value=somewhere on universe
 2                                 column=employee:department, timestamp=1523272304551, value=Development
 2                                 column=employee:designation, timestamp=1523272304551, value=engineer
 2                                 column=employee:empName, timestamp=1523272304551, value=John Doe
 2                                 column=employee:phoneNo, timestamp=1523272304551, value=987987999
 3                                 column=employee:address, timestamp=1523272304551, value=somewhere
 3                                 column=employee:department, timestamp=1523272304551, value=Development
 3                                 column=employee:designation, timestamp=1523272304551, value=engineer
 3                                 column=employee:empName, timestamp=1523272304551, value=Smith
 3                                 column=employee:phoneNo, timestamp=1523272304551, value=5432123456
 4                                 column=employee:address, timestamp=1523272304551, value=under the hood
 4                                 column=employee:department, timestamp=1523272304551, value=Development
 4                                 column=employee:designation, timestamp=1523272304551, value=manager
 4                                 column=employee:empName, timestamp=1523272304551, value=Andrew
 4                                 column=employee:phoneNo, timestamp=1523272304551, value=331222345
 5                                 column=employee:address, timestamp=1523272304551, value=nowhere
 5                                 column=employee:department, timestamp=1523272304551, value=Business
 5                                 column=employee:designation, timestamp=1523272304551, value=director
 5                                 column=employee:empName, timestamp=1523272304551, value=Troy
 5                                 column=employee:phoneNo, timestamp=1523272304551, value=9102385731
5 row(s) in 0.7250 seconds

hbase(main):002:0>
```

Data in the HBase table info table

This output now confirms that the data has been successfully imported to the HBase table `info`.

Summary

Data is growing exponentially and it is vital for any company to keep all of its data while maintaining a bare minimum cost level. In this chapter, we started by learning how to install MySQL in a Linux environment. Then, we created a database with some tables and data in it. This was done to create an initial environment in order to learn how to migrate data from a relational database such as MySQL into a Hadoop ecosystem such as HBase. We then studied the structure of the tables in the MySQL database and analyzed the data available in it.

Then, we installed HBase in our Hadoop cluster, which we set up earlier in this book. We also then performed some required configuration before proceeding with different operations in HBase. We then created the tables in it that will hold the imported data from the MySQL database.

We then learned about Sqoop, which is an open source application that makes it easier to import and export data between relational databases and Hadoop ecosystems. First, we installed Sqoop, and then we performed the required configuration. We then learned about the different parameters used in the import process of Sqoop for better understanding and clarity. Afterwards, we performed the actual import from MySQL tables and placed the data in HBase tables. Finally, we attempted to verify the availability of the data in our HBase tables, and were successful.

In the next chapter, we will learn about unstructured data and how to manipulate it with the help of different open source tools.

16
Unstructured Data

If a form of data doesn't have a predefined structure, then it falls under the classification of unstructured data. These types of data are normally huge in size compared to structured data as structured data is mainly in text form. The following are some of the examples of data that can be classified under the umbrella of unstructured data:

- Audio file
- Video file
- Document file, such as Word, PDF, and Excel
- Images
- Emails
- Text and log files
- Sensors and IoT data

Now, keep all these types of data in your mind and think of a possible solution for storing the data in such a way that you can retrieve the required information and process it using traditional systems. Take your time. Most likely you will not be able to come up with a solution, because traditional systems are not designed to deal with unstructured data effectively.

With the evolution of the big data world, more and more technological advancement is in place. These advancements lead to many possible solutions that can help us deal with unstructured data and manipulate them as per our requirements.

In the previous chapter, we learned how to import structured data from any relational database into Hadoop. In this chapter we will now import unstructured data into Hadoop and extract meaningful information from them. We will be covering the following topics:

- Importing unstructured data into Hadoop Cluster
- Converting images into text

In the subsequent sections, we will also perform a practical example to import web server log data into Hadoop. It will help to consume and store more and more data with low-cost commodity hardware using Apache Flume.

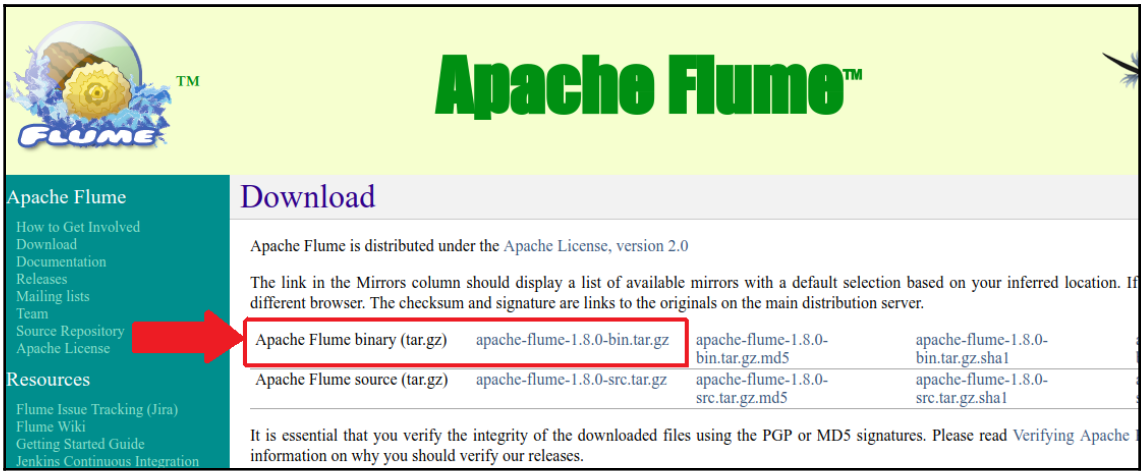
Moving data into Hadoop

We have already developed our understanding regarding Apache Flume in `Chapter 3`, *Hadoop Ecosystem*. If we recall, Apache Flume is a framework that helps move large amounts of streaming data from one place to another. It is primarily designed for log collection and aggregation from across different servers into a centralized place like Hadoop for processing and analysis. But its usage is not limited to just log aggregation. The data source connectors are customized and can be used to transport large amounts of event generated data, such as network traffic data, social media generated data, and so on; almost any type of data source.

Let's now move our discussion to the practical part, where we set up Flume to get data from any server and place it in the Hadoop directory structure.

Downloading Flume

It's time to get your hands dirty by performing the actual configuration of Apache Flume. For that, point your favorite browser to `https://flume.apache.org/download.html` to download the latest version of Apache Flume. While writing this, the current stable version of Flume is 1.8.0. Click on the Download binary link that is available on the page, as shown in the following screenshot:



Apache Flume

How to Get Involved
Download
Documentation
Releases
Mailing lists
Team
Source Repository
Apache License

Resources

Flume Issue Tracking (Jira)
Flume Wiki
Getting Started Guide
Jenkins Continuous Integration

Download

Apache Flume is distributed under the Apache License, version 2.0

The link in the Mirrors column should display a list of available mirrors with a default selection based on your inferred location. If
different browser. The checksum and signature are links to the originals on the main distribution server.

Apache Flume binary (tar.gz)	apache-flume-1.8.0-bin.tar.gz	apache-flume-1.8.0-bin.tar.gz.md5	apache-flume-1.8.0-bin.tar.gz.sha1
Apache Flume source (tar.gz)	apache-flume-1.8.0-src.tar.gz	apache-flume-1.8.0-src.tar.gz.md5	apache-flume-1.8.0-src.tar.gz.sha1

It is essential that you verify the integrity of the downloaded files using the PGP or MD5 signatures. Please read Verifying Apache
information on why you should verify our releases.

Apache Flume Download Link

Once you download the file, copy it to your home directory. Then, fire up the terminal and execute the following command to extract it:

```
$ cd ~/
$ tar -zxvf apache-flume-1.8.0-bin.tar.gz
```

Now that we have downloaded and extract Apache Flume into our home directory, which is `/home/hadoopadmin/`, we will now move forward to do path configuration.

Environment configuration

Now, we will define the Apache Flume home location in the environment path. In order to do so, edit the `bashrc` file by executing the following command:

```
$ sudo nano ~/.bashrc
```

Once it opens, go to the end of the file and write the following:

```
#Apache Flume
export FLUME_HOME="/home/hadoopadmin/apache-flume-1.8.0-bin"
export PATH=$PATH:$FLUME_HOME/bin
```

Notice that we have used the `/home/hadoopadmin/` directory, as we have placed the extracted files of Apache Flume. Make sure to change it as per your directory structure. Now save and exit the editor and reload the `bashrc` file by executing the following command, in order for the changes to take effect:

```
$ source ~/.bashrc
```

We are now ready to configure an agent that will define the source, sink, and channel to use in Flume.

Configuring agent and sink

Create a new file and write down the following code in it. In our case, we will save it with the name `flume-agent.conf`. In the following configuration code, the first word `agent` in every line of the configuration is not fixed or predefined, it can be anything, such as `logagent`, `logwritter`, `eventagent`, and so on. It will be used while invoking Apache Flume to identify that this is the agent name and that it possesses the defined properties. In the following section, we have defined the following three properties related to this agent:

- **Sources**: It contains the attributes for the sources from which this flume agent will receive the data. It can be one or multiple.
- **Sinks**: It contains the attributes for where the data will go once the flume agent receives it.
- **Channels**: It contains information regarding how the data will flow from a source to a sink, whether it will be through memory, file, JDBC, or another means.

Up till now, our configuration file looks as follows:

```
agent.sources=netcatSource
agent.sinks=hdfsWriter
agent.channels=memoryChannel
```

What we have defined here are just names that will have further properties linked to them that will make sense of actually what they are and what they will do. Like in sources, we have defined `netcatSource`; you can change it to anything you link. You will then use the same name in the following configurations to refer to it. Now that we have defined the names of the source, sink, and channel, it's time to define the individual properties of each of them.

In this example, we will bind a port in our computer to keep listening. Any data received in that port will be used as source data. For this purpose, we have following three properties:

- **Type**: It will be `netcat`, as it is a utility that is used to listen to a port and manipulate the data it receives.
- **Bind**: We are running the whole example in the same computer, so it will be `127.0.0.1`. If you are going to use a network environment, then replace it with the IP address of the computer.
- **Port**: It can be any port that is available to use. In this example we will be using `22222`.

The configuration code of our source is as follows:

```
agent.sources.netcatSource.type=netcat
agent.sources.netcatSource.bind=127.0.0.1
agent.sources.netcatSource.port=22222
```

We now move on to define the properties of our sink. In this example, we will be receiving data from our source and writing it to the HDFS file system. We will have the following properties for this type of sink:

- `type`: It defines what type of sink it is. In this example, it is HDFS, as it will write the data in the Hadoop file system.
- `hdfs.path`: It defines the path of where you want to place the files in the HDFS system.
- `rollInterval`: Number of seconds to wait to roll the received contents to a new file.
- `writeFormat`: In our case, it is Text, as we know that we are setting it up to receive text data. Other type can be writable if the data is coming in other forms.
- `hdfs.fileType`: In this example, we are using DataStream, which means it will write the data as it is in the file without compression or converting it to a sequence file.

The code of the sink configuration is as follows:

```
agent.sinks.hdfsWriter.type=hdfs
agent.sinks.hdfsWriter.hdfs.path=hdfs://localhost:9000/flumeData/
agent.sinks.hdfsWriter.rollInterval=60
agent.sinks.hdfsWriter.hdfs.writeFormat=Text
agent.sinks.hdfsWriter.hdfs.fileType=DataStream
```

Now that we have defined the source and sink and their attributes, it's now time to define the properties related to channel. In our example, we will be using a memory channel, which has only two properties:

- `type`: It contains the type of the channel – either memory, JDBC connection, or file system. In our case, it is memory.
- `capacity`: It contains the number of events that it can hold.

The code of the channel configuration is as follows:

```
agent.channels.memoryChannel.type=memory
agent.channels.memoryChannel.capacity=10000
```

One last configuration to define is that which links the source and the sink to the memory channel. It can be done as follows:

```
agent.sources.netcatSource.channels=memoryChannel
agent.sinks.hdfsWriter.channel=memoryChannel
```

The complete code of the configuration file is as follows:

```
agent.sources=netcatSource
agent.sinks=hdfsWriter
agent.channels=memoryChannel

agent.sources.netcatSource.type=netcat
agent.sources.netcatSource.bind=127.0.0.1
agent.sources.netcatSource.port=22222

agent.sinks.hdfsWriter.type=hdfs
agent.sinks.hdfsWriter.hdfs.path=hdfs://localhost:9000/flumeData/
agent.sinks.hdfsWriter.rollInterval=60
agent.sinks.hdfsWriter.hdfs.writeFormat=Text
agent.sinks.hdfsWriter.hdfs.fileType=DataStream

agent.channels.memoryChannel.type=memory
agent.channels.memoryChannel.capacity=10000

agent.sources.netcatSource.channels=memoryChannel
agent.sinks.hdfsWriter.channel=memoryChannel
```

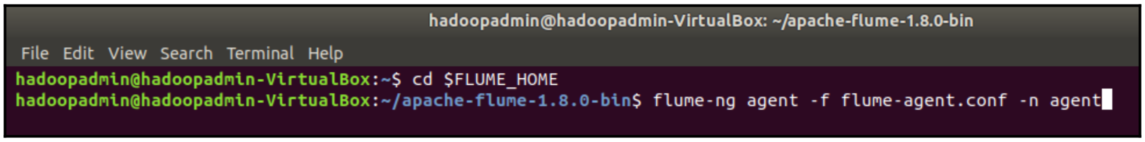
Now that we have defined all the configuration related to a Flume agent, it's time to save this file in your Apache Flume installation folder, which in our case is `/home/hadoopadmin/apache-flume-1.8.0-bin`. The filename we will be using is `flume-agent.conf`, although of course you can name it anything you want.

Running Apache Flume

It's now time to start our agent. It can be done by executing the following commands:

```
$ cd $FLUME_HOME
$ flume-ng agent -f flume-agent.conf -n agent
```

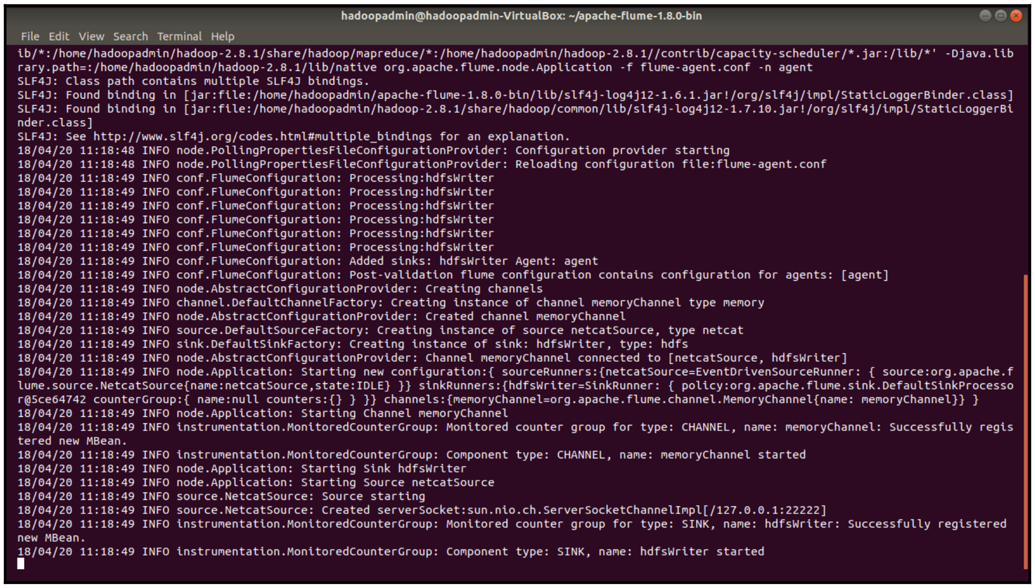
This is done as shown in the following screenshot:


hadoopadmin@hadoopadmin-VirtualBox: ~/apache-flume-1.8.0-bin
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~$ cd $FLUME_HOME
hadoopadmin@hadoopadmin-VirtualBox:~/apache-flume-1.8.0-bin$ flume-ng agent -f flume-agent.conf -n agent
```

Apache Flume agent command to start an agent

Once you execute it, it will display series of configuration outputs, as shown in the following screenshot:


hadoopadmin@hadoopadmin-VirtualBox: ~/apache-flume-1.8.0-bin
File Edit View Search Terminal Help

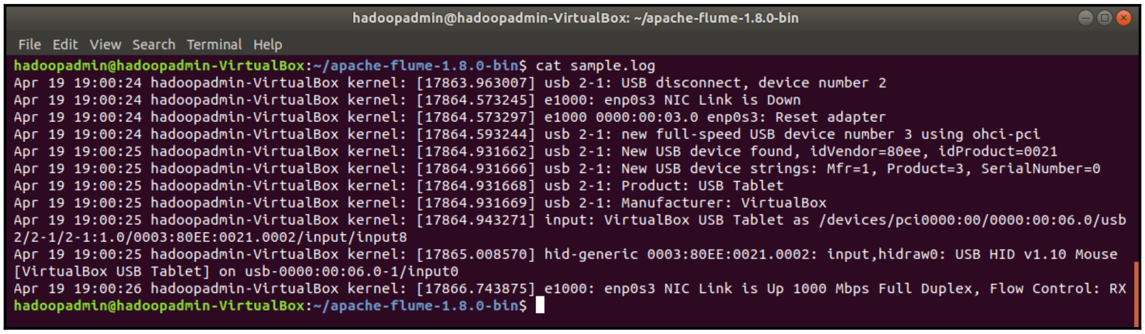
```
ib/*:/home/hadoopadmin/hadoop-2.8.1/share/hadoop/mapreduce/*:/home/hadoopadmin/hadoop-2.8.1//contrib/capacity-scheduler/*.jar:/lib/*' -Djava.lib
rary.path=:/home/hadoopadmin/hadoop-2.8.1/lib/native org.apache.flume.node.Application -f flume-agent.conf -n agent
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoopadmin/apache-flume-1.8.0-bin/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoopadmin/hadoop-2.8.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBi
nder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
18/04/20 11:18:48 INFO node.PollingPropertiesFileConfigurationProvider: Configuration provider starting
18/04/20 11:18:48 INFO node.PollingPropertiesFileConfigurationProvider: Reloading configuration file:flume-agent.conf
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Processing:hdfsWriter
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Added sinks: hdfsWriter Agent: agent
18/04/20 11:18:49 INFO conf.FlumeConfiguration: Post-validation flume configuration contains configuration for agents: [agent]
18/04/20 11:18:49 INFO node.AbstractConfigurationProvider: Creating channels
18/04/20 11:18:49 INFO channel.DefaultChannelFactory: Creating instance of channel memoryChannel type memory
18/04/20 11:18:49 INFO node.AbstractConfigurationProvider: Created channel memoryChannel
18/04/20 11:18:49 INFO source.DefaultSourceFactory: Creating instance of source netcatSource, type netcat
18/04/20 11:18:49 INFO sink.DefaultSinkFactory: Creating instance of sink: hdfsWriter, type: hdfs
18/04/20 11:18:49 INFO node.AbstractConfigurationProvider: Channel memoryChannel connected to [netcatSource, hdfsWriter]
18/04/20 11:18:49 INFO node.Application: Starting new configuration:{ sourceRunners:{netcatSource=EventDrivenSourceRunner: { source:org.apache.f
lume.source.NetcatSource{name:netcatSource,state:IDLE} }} sinkRunners:{hdfsWriter=SinkRunner: { policy:org.apache.flume.sink.DefaultSinkProcesso
r@5ce64742 counterGroup:{ name:null counters:{} } }} channels:{memoryChannel=org.apache.flume.channel.MemoryChannel{name: memoryChannel}} }
18/04/20 11:18:49 INFO node.Application: Starting Channel memoryChannel
18/04/20 11:18:49 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: CHANNEL, name: memoryChannel: Successfully regis
tered new MBean.
18/04/20 11:18:49 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: memoryChannel started
18/04/20 11:18:49 INFO node.Application: Starting Sink hdfsWriter
18/04/20 11:18:49 INFO node.Application: Starting Source netcatSource
18/04/20 11:18:49 INFO source.NetcatSource: Source starting
18/04/20 11:18:49 INFO source.NetcatSource: Created serverSocket:sun.nio.ch.ServerSocketChannelImpl[/127.0.0.1:22222]
18/04/20 11:18:49 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SINK, name: hdfsWriter: Successfully registered
new MBean.
18/04/20 11:18:49 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: hdfsWriter started
```

Apache Flume agent output

The last three lines of this output confirms that the source, sink, and channel have successfully started.

Transferring a log file

Now that we have successfully confirmed and started our flume agent, it's now time to take a test drive to check it from end to end. For this purpose, we have created a sample log file in the `$FLUME_HOME` path. The following screenshot shows the log file and its contents:



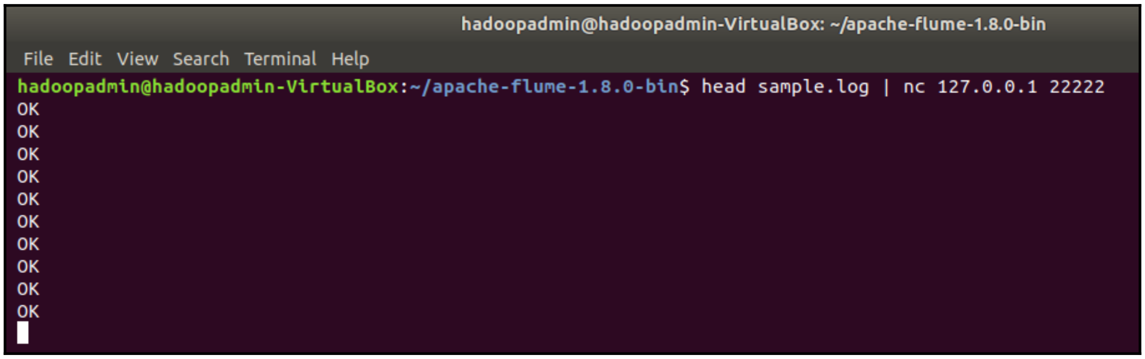
```
hadoopadmin@hadoopadmin-VirtualBox:~/apache-flume-1.8.0-bin$ cat sample.log
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17863.963007] usb 2-1: USB disconnect, device number 2
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.573245] e1000: enp0s3 NIC Link is Down
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.573297] e1000 0000:00:03.0 enp0s3: Reset adapter
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.593244] usb 2-1: new full-speed USB device number 3 using ohci-pci
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931662] usb 2-1: New USB device found, idVendor=80ee, idProduct=0021
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931666] usb 2-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931668] usb 2-1: Product: USB Tablet
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931669] usb 2-1: Manufacturer: VirtualBox
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.943271] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb
2/2-1/2-1:1.0/0003:80EE:0021.0002/input/input8
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17865.008570] hid-generic 0003:80EE:0021.0002: input,hidraw0: USB HID v1.10 Mouse
[VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
Apr 19 19:00:26 hadoopadmin-VirtualBox kernel: [17866.743875] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
hadoopadmin@hadoopadmin-VirtualBox:~/apache-flume-1.8.0-bin$
```

Sample.log file contents

We will now send this log file to the Flume source, which is `netcat` running on `127.0.0.1` port `22222`. In order to do so, execute the following command:

```
$ head sample.log | nc 127.0.0.1 22222
```

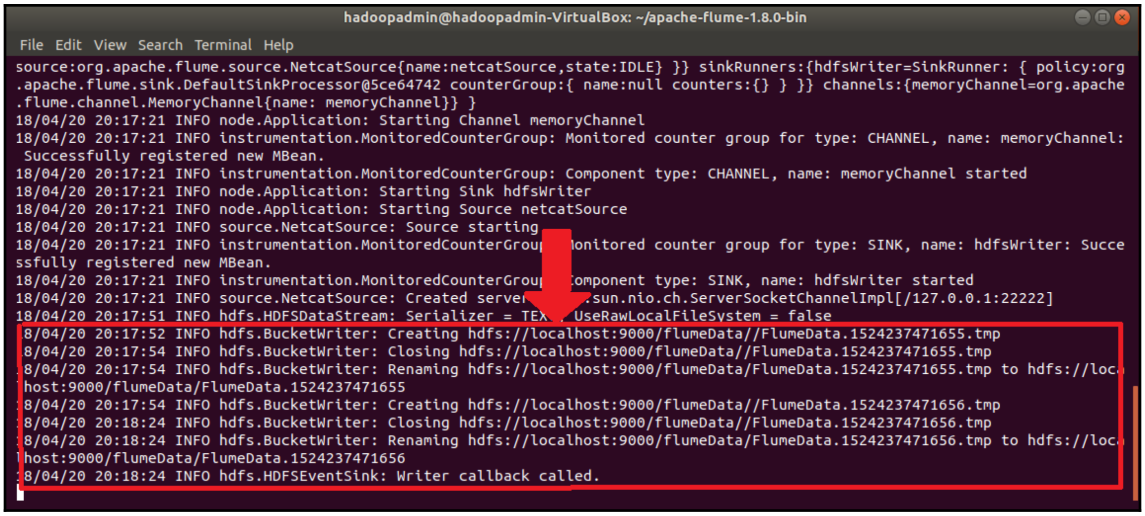
The following screenshot displays the output of this command:



```
hadoopadmin@hadoopadmin-VirtualBox:~/apache-flume-1.8.0-bin$ head sample.log | nc 127.0.0.1 22222
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
```

Head command execution output

In its output, the series of `OK` represents the contents of the file transferred as per the buffered size available. Now if you look into the terminal window where you started the Flume agent, it displays the creation of the file in the HDFS system, as shown in the following screenshot:



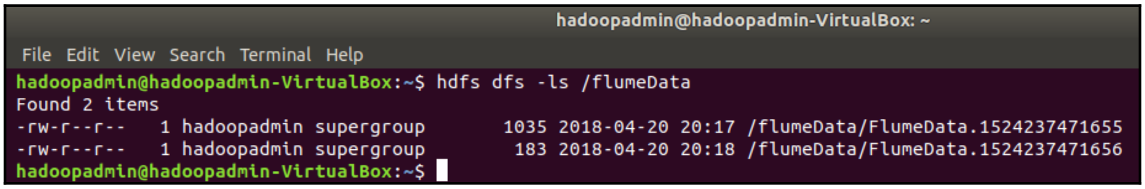
```
source:org.apache.flume.source.NetcatSource{name:netcatSource,state:IDLE} }} sinkRunners:{hdfsWriter=SinkRunner: { policy:org
.apache.flume.sink.DefaultSinkProcessor@5ce64742 counterGroup:{ name:null counters:{} } }} channels:{memoryChannel=org.apache
.flume.channel.MemoryChannel{name: memoryChannel}} }
18/04/20 20:17:21 INFO node.Application: Starting Channel memoryChannel
18/04/20 20:17:21 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: CHANNEL, name: memoryChannel:
 Successfully registered new MBean.
18/04/20 20:17:21 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: memoryChannel started
18/04/20 20:17:21 INFO node.Application: Starting Sink hdfsWriter
18/04/20 20:17:21 INFO node.Application: Starting Source netcatSource
18/04/20 20:17:21 INFO source.NetcatSource: Source starting
18/04/20 20:17:21 INFO instrumentation.MonitoredCounterGrou   onitored counter group for type: SINK, name: hdfsWriter: Succe
ssfully registered new MBean.
18/04/20 20:17:21 INFO instrumentation.MonitoredCounterGrou   omponent type: SINK, name: hdfsWriter started
18/04/20 20:17:21 INFO source.NetcatSource: Created serve    sun.nio.ch.ServerSocketChannelImpl[/127.0.0.1:22222]
18/04/20 20:17:51 INFO hdfs.HDFSDataStream: Serializer = TEX   UseRawLocalFileSystem = false
8/04/20 20:17:52 INFO hdfs.BucketWriter: Creating hdfs://localhost:9000/flumeData//FlumeData.1524237471655.tmp
8/04/20 20:17:54 INFO hdfs.BucketWriter: Closing hdfs://localhost:9000/flumeData//FlumeData.1524237471655.tmp
8/04/20 20:17:54 INFO hdfs.BucketWriter: Renaming hdfs://localhost:9000/flumeData/FlumeData.1524237471655.tmp to hdfs://loc
host:9000/flumeData/FlumeData.1524237471655
8/04/20 20:17:54 INFO hdfs.BucketWriter: Creating hdfs://localhost:9000/flumeData//FlumeData.1524237471656.tmp
8/04/20 20:18:24 INFO hdfs.BucketWriter: Closing hdfs://localhost:9000/flumeData//FlumeData.1524237471656.tmp
8/04/20 20:18:24 INFO hdfs.BucketWriter: Renaming hdfs://localhost:9000/flumeData/FlumeData.1524237471656.tmp to hdfs://loc
host:9000/flumeData/FlumeData.1524237471656
8/04/20 20:18:24 INFO hdfs.HDFSEventSink: Writer callback called.
```

Flume agent log generation in the terminal window

One thing to note here is that it has created two files. This is because while configuring the agent, we defined that the capacity of it, once exceeded, will create a new file. Now, to confirm that the content has been successfully transferred to the `hdfs` file system, execute the following command first to list the file created in the file system:

```
$ hdfs dfs -ls /flumeData
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

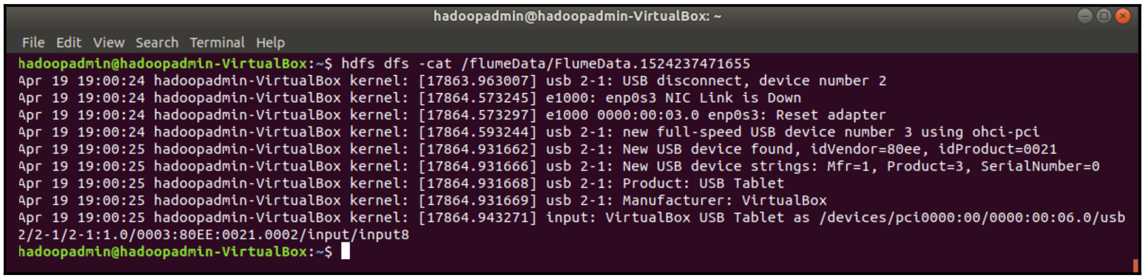
```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -ls /flumeData
Found 2 items
-rw-r--r--   1 hadoopadmin supergroup       1035 2018-04-20 20:17 /flumeData/FlumeData.1524237471655
-rw-r--r--   1 hadoopadmin supergroup        183 2018-04-20 20:18 /flumeData/FlumeData.1524237471656
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Files available in Flume data directory in HDFS

Execute the following command to view the contents of the first file, just to verify that the contents of the sample log file have been successfully transferred:

```
$ hdfs dfs -cat /flumeData/FlumeData.1524237471655
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

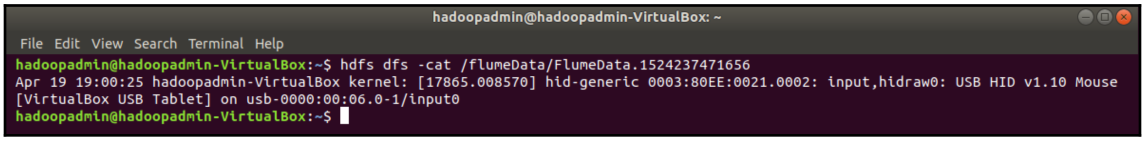
```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -cat /flumeData/FlumeData.1524237471655
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17863.963007] usb 2-1: USB disconnect, device number 2
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.573245] e1000: enp0s3 NIC Link is Down
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.573297] e1000 0000:00:03.0 enp0s3: Reset adapter
Apr 19 19:00:24 hadoopadmin-VirtualBox kernel: [17864.593244] usb 2-1: new full-speed USB device number 3 using ohci-pci
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931662] usb 2-1: New USB device found, idVendor=80ee, idProduct=0021
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931666] usb 2-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931668] usb 2-1: Product: USB Tablet
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.931669] usb 2-1: Manufacturer: VirtualBox
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17864.943271] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb
2/2-1/2-1:1.0/0003:80EE:0021.0002/input/input8
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Content of the first file created by Flume

Execute the following command to view the contents of the second file:

```
$ hdfs dfs -cat /flumeData/FlumeData.1524237471656
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~$ hdfs dfs -cat /flumeData/FlumeData.1524237471656
Apr 19 19:00:25 hadoopadmin-VirtualBox kernel: [17865.008570] hid-generic 0003:80EE:0021.0002: input,hidraw0: USB HID v1.10 Mouse
[VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Content of the second file create by Flume

It confirms that the content of the sample log file has successfully transferred to HDFS using Apache Flume. You can tune up the agent configuration file as per your requirement to make Apache Flume more effective.

We will now learn how to convert an image into text; this will in turn help us to extract meaningful information from one of unstructured data types.

Converting images into text for analysis

In today's world, most of the information is available on the internet. This information can be in the form of text, videos, or images. We have many tools and techniques available for analyzing text and extract relevant information from it, but what about a scanned image from a book, or text that is available in the form of an image?

Tesseract OCR

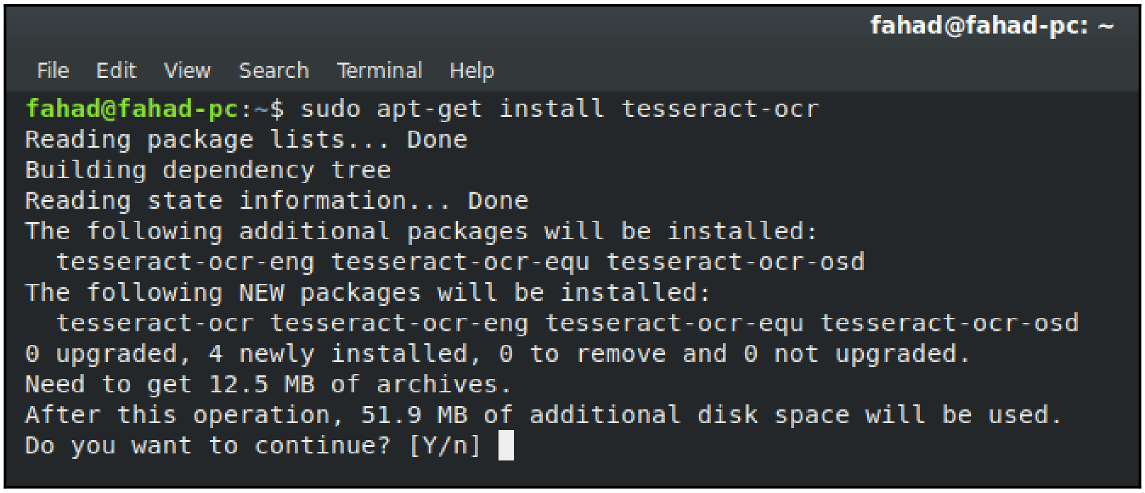
Here, we will now learn how to extract text from an image. For this purpose, we will use the Tesseract framework. The Tesseract **OCR** (**optical character recognition**) project is sponsored by Google, and is available as an open source project under the Apache 2.0 license. It is capable of converting images to text in different languages. It can also be trained to learn new languages.

Installing Tesseract

The first step is to install Tesseract in Ubuntu by executing the following command:

```
$ sudo apt-get install tesseract-ocr
```

The following screenshot displays the output of this command:



```
fahad@fahad-pc:~$ sudo apt-get install tesseract-ocr
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  tesseract-ocr-eng tesseract-ocr-equ tesseract-ocr-osd
The following NEW packages will be installed:
  tesseract-ocr tesseract-ocr-eng tesseract-ocr-equ tesseract-ocr-osd
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 12.5 MB of archives.
After this operation, 51.9 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Tesseract installation

Once it is executed successfully, it will install the Tesseract framework on your local machine.

Practical example

Now that we have set up the Tesseract framework, it's time to start writing the code. Create a new text file and save it with the name `imgToText.py`. For now, we will go through the core concepts used in this program for better understanding. Then, in the next section, we will provide the full code of the program with some basic validation and conditions.

At the beginning of the program, the first condition that we will check is whether Tesseract is installed in the local machine, so that we can correctly warn the end user to install it first before proceeding further, as it is the core OCR engine that will convert the image into text. For this purpose, the following code is required:

```
if (call(['which', 'tesseract'])):
        print("Tesseract is not installed. Please install it first")
        print("If you are in Linux environment, please execute the")
print("following command in terminal in order to install it")
        print("sudo apt-get install tesseract-ocr")
        sys.exit(1)
```

The program actually tries to call a Linux command, which supposedly returns the path of the application installed. If the program doesn't find the Tesseract, it will enter into this `if` condition, which will display a message along with the Linux command to install Tesseract and then exit the program. This is the best possible way to check the dependencies within your program and also help the end user to resolve any issues encountered.

In order to execute a separate process from within your Python script, as we did in this code while defining the if condition, you are required to import it by mentioning the following line at the start of your script:

```
from subprocess import call
```

In our program, we will accept two arguments. The first one is a mandatory argument, which will specify the image file that we will convert into text. The second argument will specify in which local directory to place the converted text. To accept the argument in our program, we will import a helper class, `argparse`, in our program by specifying the following code in the first section of the program, where all the other imports are defined:

```
import argparse
```

The next step is to define a variable that will initiate the parser. This is done using the following code:

```
parse = argparse.ArgumentParser()
```

We will now define the first argument that will accept the image filename, along with the help text for the end user to better understand what is required from them:

```
parse.add_argument("img", help="Image filename is required to convert it to
text")
```

Similarly, the following code is used to define the second argument:

```
parse.add_argument("outPath", nargs="?", help="(optional) Path to store
converted Text from image, or else it will place the text in the same
directory")
```

This argument is optional; if the user specifies this then the output will be generated in that folder, otherwise it will generate the output in the same folder as the input image.

We will now generate the output filename as we did with the input filename, along with the `"-text"` postfix. For this purpose, the following code will be used:

```
 textFileName = os.path.splitext(img)[0] + "-text"
```

It's now time to initiate the conversion process on the image that we got as an input argument. We will now call a subprocess as we did before to call the Tesseract framework with the required arguments. It takes two arguments, the input image and the output image. We will use the following code to execute the subprocess:

```
call(["tesseract", img, textFileName], stdout=STDNULL)
```

Once this code gets executed, if everything works fine then you will see an output file in the location you specified with the input argument, or by default, in the same folder as the input file.

This is the brief explanation of the main components of the program. In between the explained code, there are multiple checks and conditions that were skipped in the explanation as they are basic in nature. The complete code of the program is available in the subsequent section.

Complete code

The complete code of the program to convert an image into text is as follows. Make sure to keep the indent to make sure the Python program executes as expected:

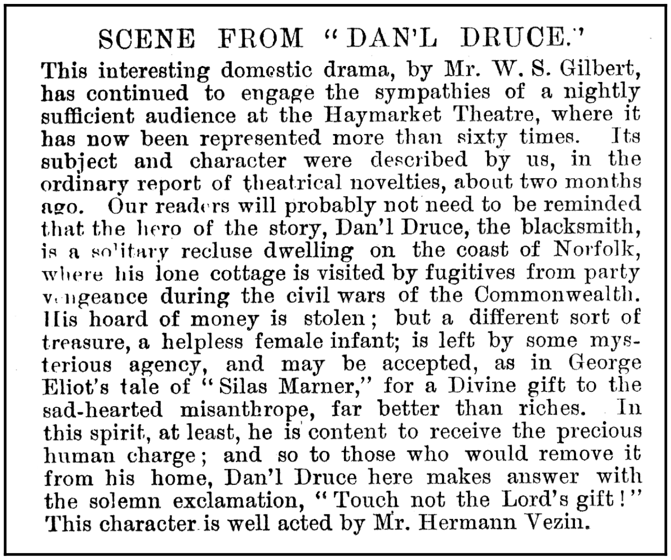
```
import sys
import os
import argparse
from subprocess import call

STDNULL = open(os.devnull, 'w')

def main(img, outPath):
    print("=============================================================")
    print("BigData Architect's Handbook Example: Convert Image into text")
print("=============================================================\n")
    # Checking whether Tesseract Library is installed in the system
    print("Checking whether tesseract library is installed ...")
    if (call(['which', 'tesseract'])):
        print("Tesseract is not installed. Please install it first")
        print("If you are in Linux environment, please execute the
following command in terminal in order to install it")
        print("sudo apt-get install tesseract-ocr")
        sys.exit(1)
    else:
        print("It is installed on the above mentioned path")
        print("------------------------------------------\n")
        # Absolute path of the directory where the image is placed
        img = os.path.abspath(img)
        inputDirectory = os.path.dirname(img)
        print("Working Directory \t: " + inputDirectory)
        print("Image File Name \t: " + img)
        # Output filename based on the input image file with -text postfix
        textFileName = os.path.splitext(img)[0] + "-text"
        if (outPath is not None):
            # Get the absolute path of the output directory
            outputDirectory = os.path.abspath(outPath)

            # check whether the output directory exist or not
            if (not os.path.exists(outputDirectory)):
                # if not, create the directory
                os.makedirs(outputDirectory)

            # Generate the complete name along with absolute path of the
generated text file
            textFileName = os.path.join(outputDirectory, textFileName)
        else:
            textFileName = os.path.join(inputDirectory, textFileName)
        print("Output Filename \t: " + textFileName +"\n")

        # Convert the image into Text
        call(["tesseract", img, textFileName], stdout=STDNULL)

        if (os.path.exists(textFileName + ".txt")):
            print("Operation completed and the output text has been
generated")
        else:
            print("Something went wrong. Please rerun the process")


if __name__ == '__main__':
    parse = argparse.ArgumentParser()
    parse.add_argument("img", help="Image filename is required to convert
it to text")
    parse.add_argument("outPath", nargs="?", help="(optional) Path to store
converted Text from image, \
                 or else it will place the text in the same directory")
    args = parse.parse_args()
    main(args.img, args.outPath)
```

You can download this code from the Github repository.

Program execution

We will now execute this program to test if it is working fine or not, and whether it is able to convert an image into text. For this purpose, we have placed the code file in the `ImageToText` directory, which is in our home directory. The complete path is `/home/hadoopadmin/ImageToText/imgToText.py`.

We will be using the following photo to convert it into text. Place this image file in the same directory as the program code, with the name `sample.png`:



SCENE FROM "DAN'L DRUCE."

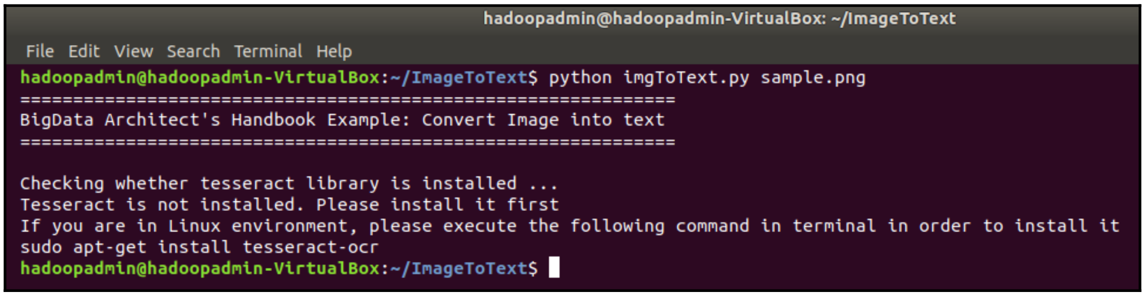
This interesting domestic drama, by Mr. W. S. Gilbert, has continued to engage the sympathies of a nightly sufficient audience at the Haymarket Theatre, where it has now been represented more than sixty times. Its subject and character were described by us, in the ordinary report of theatrical novelties, about two months ago. Our readers will probably not need to be reminded that the hero of the story, Dan'l Druce, the blacksmith, is a solitary recluse dwelling on the coast of Norfolk, where his lone cottage is visited by fugitives from party vengeance during the civil wars of the Commonwealth. His hoard of money is stolen; but a different sort of treasure, a helpless female infant; is left by some mysterious agency, and may be accepted, as in George Eliot's tale of "Silas Marner," for a Divine gift to the sad-hearted misanthrope, far better than riches. In this spirit, at least, he is content to receive the precious human charge; and so to those who would remove it from his home, Dan'l Druce here makes answer with the solemn exclamation, "Touch not the Lord's gift!" This character is well acted by Mr. Hermann Vezin.

Sample image to convert into text

We are now ready to execute the program to convert this image into text. First, we will simulate considering the Tesseract to be missing. For this purpose, fire up the terminal and execute the following program:

```
$ cd ~/ImageToText/
$ python imgToText.py sample.png
```

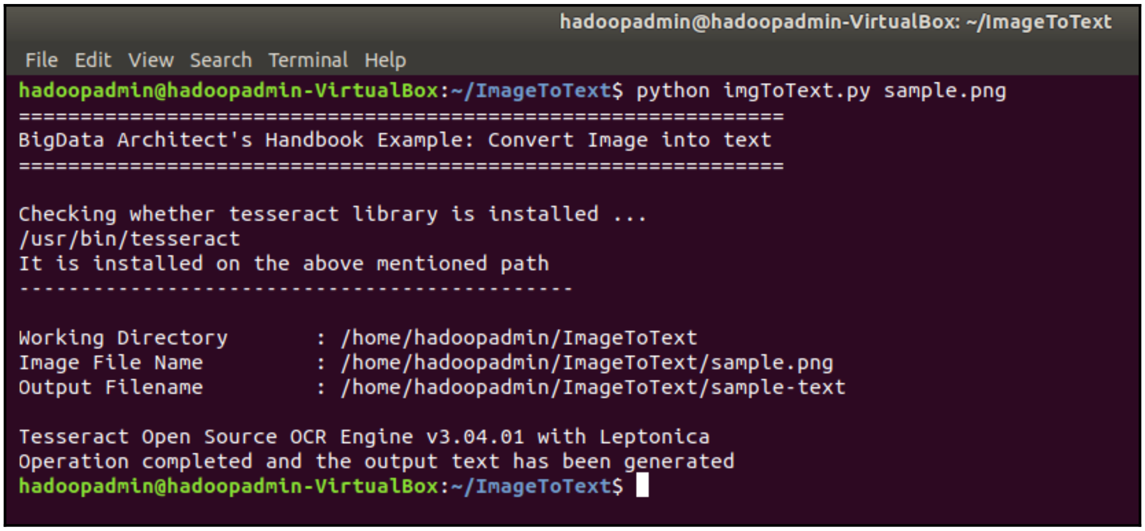
The following screenshot displays the output of the program:


hadoopadmin@hadoopadmin-VirtualBox: ~/ImageToText
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$ python imgToText.py sample.png
==============================================================
BigData Architect's Handbook Example: Convert Image into text
==============================================================

Checking whether tesseract library is installed ...
Tesseract is not installed. Please install it first
If you are in Linux environment, please execute the following command in terminal in order to install it
sudo apt-get install tesseract-ocr
hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$
```

imgToText Program output if Tesseract is missing

We will now consider the case that we have installed Tesseract and now test the program. Execute the commands again. The following screenshot displays the output of the program:


hadoopadmin@hadoopadmin-VirtualBox: ~/ImageToText
File Edit View Search Terminal Help

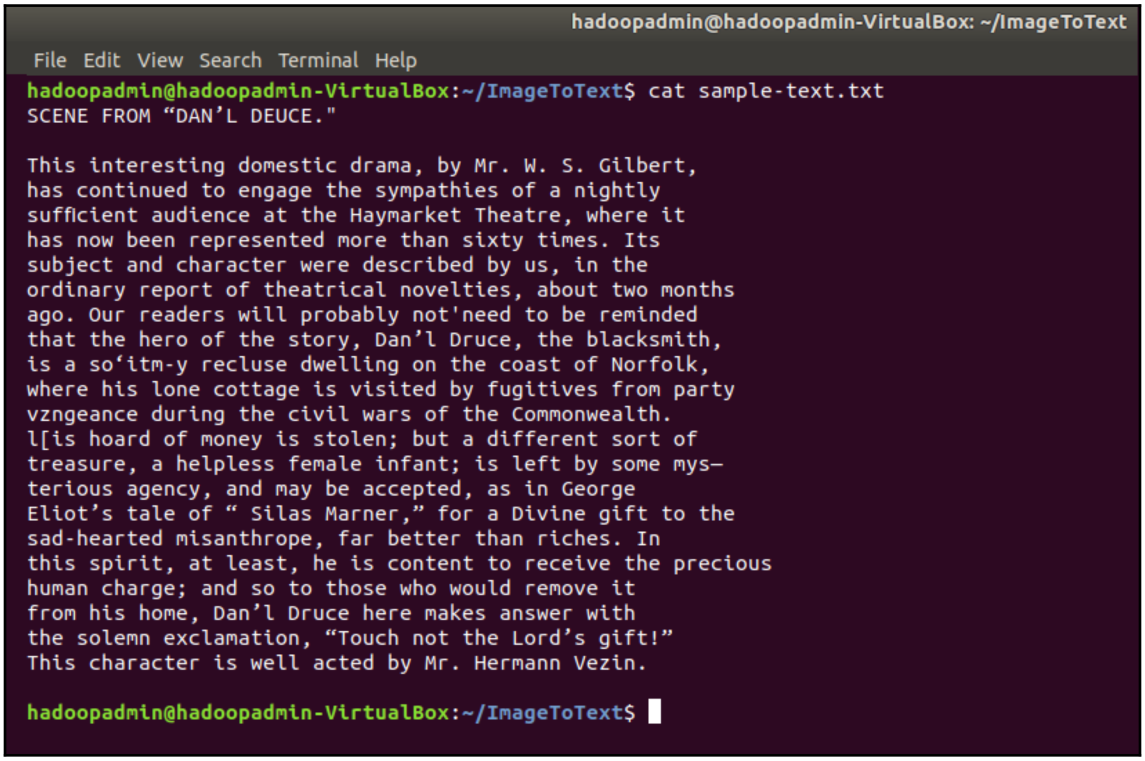
```
hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$ python imgToText.py sample.png
==============================================================
BigData Architect's Handbook Example: Convert Image into text
==============================================================

Checking whether tesseract library is installed ...
/usr/bin/tesseract
It is installed on the above mentioned path
---------------------------------------------

Working Directory        : /home/hadoopadmin/ImageToText
Image File Name          : /home/hadoopadmin/ImageToText/sample.png
Output Filename          : /home/hadoopadmin/ImageToText/sample-text

Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Operation completed and the output text has been generated
hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$
```

imgToText program output after installing Tesseract

Now that our program has successfully executed, we will now test the generated text. For this purpose, execute the following command:

```
$ cat sample-text.text
```

The following screenshot displays the output of this command:


hadoopadmin@hadoopadmin-VirtualBox: ~/ImageToText
File Edit View Search Terminal Help

```
hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$ cat sample-text.txt
SCENE FROM “DAN’L DEUCE."

This interesting domestic drama, by Mr. W. S. Gilbert,
has continued to engage the sympathies of a nightly
sufficient audience at the Haymarket Theatre, where it
has now been represented more than sixty times. Its
subject and character were described by us, in the
ordinary report of theatrical novelties, about two months
ago. Our readers will probably not'need to be reminded
that the hero of the story, Dan’l Druce, the blacksmith,
is a so‘itm-y recluse dwelling on the coast of Norfolk,
where his lone cottage is visited by fugitives from party
vzngeance during the civil wars of the Commonwealth.
l[is hoard of money is stolen; but a different sort of
treasure, a helpless female infant; is left by some mys—
terious agency, and may be accepted, as in George
Eliot’s tale of “ Silas Marner,” for a Divine gift to the
sad-hearted misanthrope, far better than riches. In
this spirit, at least, he is content to receive the precious
human charge; and so to those who would remove it
from his home, Dan’l Druce here makes answer with
the solemn exclamation, “Touch not the Lord’s gift!”
This character is well acted by Mr. Hermann Vezin.

hadoopadmin@hadoopadmin-VirtualBox:~/ImageToText$
```

Text generated from the imgToText program

Now, you can analyze the output of our program – how well it has converted the text from the image. Imagine the possibilities and application of this scenario where the text is in binary form as an image; now you can read and analyze it, similar to other types of text based data.

Summary

In this chapter, we learned about the data that is categorized as unstructured data. We then jumped into the practical of importing data into Hadoop cluster by using Apache Flume. Apache Flume is designed on the principle of having the source and sink connected to each other using channel. We have already discussed the theory of Apache Flume in `Chapter 3`, *Hadoop Ecosystem*. Data comes into Flume with the help of a source, travels through channels, and dumps into sinks. In our example, we configured `netcat` to be an Apache Flume source. `netcat` sends data to Flume, which will be processed in memory. Memory is served as a channel to connect a source to a sink. Channel will then pass it on to sink. We configured Hadoop file system as an Apache Flume sink so that it will save the received data into HDFS as files.

We then developed a program that converts an image into text. We went through the core functionality of the program and discussed it for better understanding. Then, we went through the full code of the program with different conditions and logic. Afterwards, we executed the program with a sample image, which the program successfully converted into text. It then used this image content to create a text file. This text file will later be used to stream the data to HDFS for further analysis and storage based on your requirements, similar to what we did in the first section of this chapter.

In the next chapter, we will learn how to use the different open source tools that are available for data visualization and analysis purposes.

17
Data Visualization

Data is a very powerful weapon. If used properly, it can lead to many victories. In the business world, *victory* means a happy customer while company remain competitive in the market. Sometimes data presented is in numerical form or in tabular form, which makes it very difficult to find a pattern or to visually analyze it.

On the other hand, if the same data is presented to you in graphical form, such as in the form of different types of charts or an interactive dashboard, most of the time it becomes easy to visualize what to look for, and it may lead to meaningful discoveries to help in a better understanding and analysis of the data.

In this chapter, we will go through the following data visualization libraries, which will help to present data in the form of different types of visual elements, such as charts:

- Matplotlib
- D3.js

Let's start with the first visualization library, Matplotlib.

Matplotlib

Matplotlib is a charting library that helps to create 2D charts with a few lines of code. If you know what type of chart you want to display and you have the required data, then Matplotlib helps you to achieve it very easily and with minimal code. It is mainly available to use with Python scripts, but can also be used with IPython shells, Jupyter notebook, and web application servers.

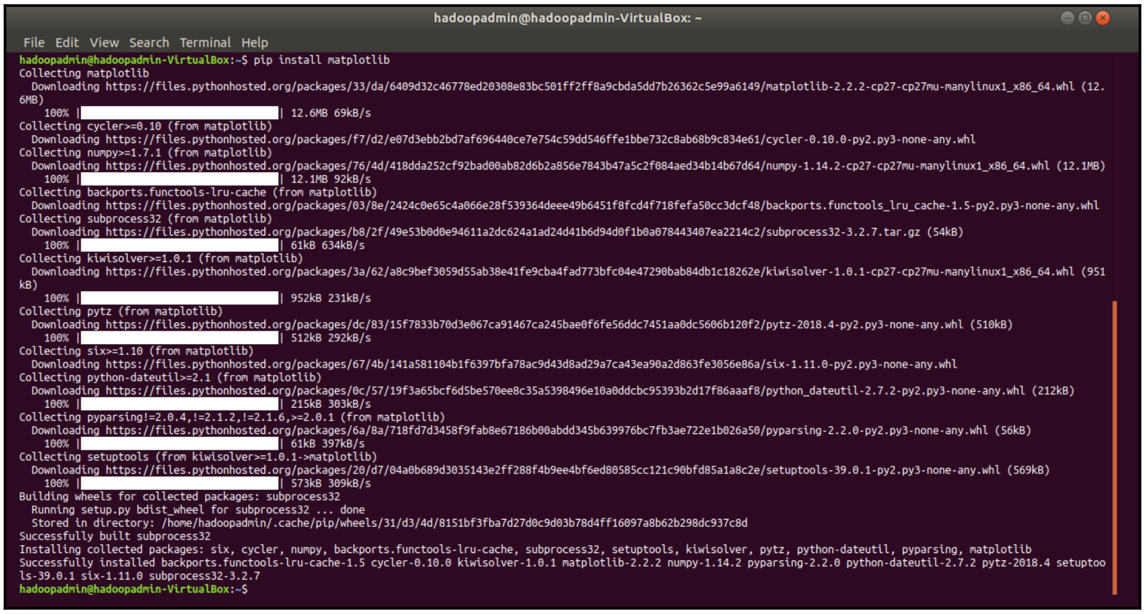
Let's move ahead and start setting up the environment to use the `matplotlib` library.

Installing Matplotlib

Installing Matplotlib is pretty simple. It is available to be installed using **Python package manager** (**Pip**). Let's start the installation process by executing the following command:

```
$ sudo pip install matplotlib
```

The following screenshot displays the output of this command:



```
hadoopadmin@hadoopadmin-VirtualBox:~$ pip install matplotlib
Collecting matplotlib
  Downloading https://files.pythonhosted.org/packages/33/da/6409d32c46778ed20308e83bc501ff2ff8a9cbda5dd7b26362c5e99a6149/matplotlib-2.2.2-cp27-cp27mu-manylinux1_x86_64.whl (12.
6MB)
    100% |████████████████████████████████| 12.6MB 69kB/s
Collecting cycler>=0.10 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c834e61/cycler-0.10.0-py2.py3-none-any.whl
Collecting numpy>=1.7.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/76/4d/418dda252cf92bad00ab82d6b2a856e7843b47a5c2f084aed34b14b67d64/numpy-1.14.2-cp27-cp27mu-manylinux1_x86_64.whl (12.1MB)
    100% |████████████████████████████████| 12.1MB 92kB/s
Collecting backports.functools-lru-cache (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/03/8e/2424c0e65c4a066e28f539364deee49b6451f8fcd4f718fefa50cc3dcf48/backports.functools_lru_cache-1.5-py2.py3-none-any.whl
Collecting subprocess32 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/b8/2f/49e53b0d0e94611a2dc624a1ad24d41b6d94d0f1b0a078443407ea2214c2/subprocess32-3.2.7.tar.gz (54kB)
    100% |████████████████████████████████| 61kB 634kB/s
Collecting kiwisolver>=1.0.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/3a/62/a8c9bef3059d55ab38e41fe9cba4fad773bfc04e47290bab84db1c18262e/kiwisolver-1.0.1-cp27-cp27mu-manylinux1_x86_64.whl (951
kB)
    100% |████████████████████████████████| 952kB 231kB/s
Collecting pytz (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/dc/83/15f7833b70d3e067ca91467ca245bae0f6fe56ddc7451aa0dc5606b120f2/pytz-2018.4-py2.py3-none-any.whl (510kB)
    100% |████████████████████████████████| 512kB 292kB/s
Collecting six>=1.10 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bfa78ac9d43d8ad29a7ca43ea90a2d863fe3056e86a/six-1.11.0-py2.py3-none-any.whl
Collecting python-dateutil>=2.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/0c/57/19f3a65bcf6d5be570ee8c35a5398496e10a0ddcbc95393b2d17f86aaaf8/python_dateutil-2.7.2-py2.py3-none-any.whl (212kB)
    100% |████████████████████████████████| 215kB 303kB/s
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/6a/8a/718fd7d3458f9fab8e67186b00abdd345b639976bc7fb3ae722e1b026a50/pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 397kB/s
Collecting setuptools (from kiwisolver>=1.0.1->matplotlib)
  Downloading https://files.pythonhosted.org/packages/20/d7/04a0b689d3035143e2ff288f4b9ee4bf6ed80585cc121c90bfd85a1a8c2e/setuptools-39.0.1-py2.py3-none-any.whl (569kB)
    100% |████████████████████████████████| 573kB 309kB/s
Building wheels for collected packages: subprocess32
  Running setup.py bdist_wheel for subprocess32 ... done
  Stored in directory: /home/hadoopadmin/.cache/pip/wheels/31/d3/4d/8151bf3fba7d27d0c9d03b78d4ff16097a8b62b298dc937c8d
Successfully built subprocess32
Installing collected packages: six, cycler, numpy, backports.functools-lru-cache, subprocess32, setuptools, kiwisolver, pytz, python-dateutil, pyparsing, matplotlib
Successfully installed backports.functools-lru-cache-1.5 cycler-0.10.0 kiwisolver-1.0.1 matplotlib-2.2.2 numpy-1.14.2 pyparsing-2.2.0 python-dateutil-2.7.2 pytz-2018.4 setuptoo
ls-39.0.1 six-1.11.0 subprocess32-3.2.7
hadoopadmin@hadoopadmin-VirtualBox:~$
```

Matplotlib installation using Pip

If this is a fresh copy of Python, then it will also download a few other dependency packages along with Matplotlib. Matplotlib also requires a python `tkinter` package, which can be installed using the following command:

```
$ sudo apt install python-tk
```

Now that we have successfully installed Matplotlib, let's start playing with it.

Line chart

We will start writing our first program in Python by importing Matplotlib using the following code:

```
import matplotlib.pyplot as plt
```

If you are familiar with charts, in a line chart, there are some values on the *x* axis and some values on the *y* axis against which the chart gets plotted. In our program, we will provide the values in the form of a matrix, as shown in the following code:

```
x = [1, 5, 3, 7]
y = [2, 5, 1, 5]
```

In real life scenarios, these values may be coming from a databases or you may get them after some analysis of the data available to you. Anyway, let's now plot them in the line chart and visualize the output it generates by using the following code:

```
plt.plot(x, y)
plt.show()
```

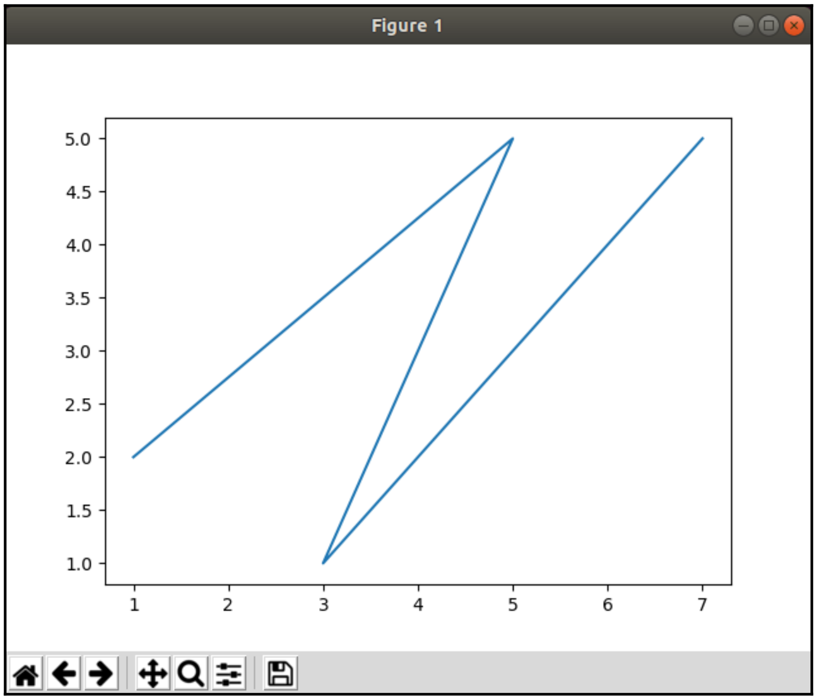
The complete code of our program now becomes the following:

```
import matplotlib.pyplot as plt

x = [1, 5, 3, 7]
y = [2, 5, 1, 5]

plt.plot(x, y)
plt.show()
```

Create a folder named `Matplotlib` in your home folder and save the file as `linechart.py`. Now execute the following command to run it:

```
$ cd ~/Matplotlib
$ python linechart.py
```

It will generate the following:



Linechart generation using Matplotlib

We have successfully created our first chart with Linechart. Let's now move on to creating a bar chart.

Bar charts

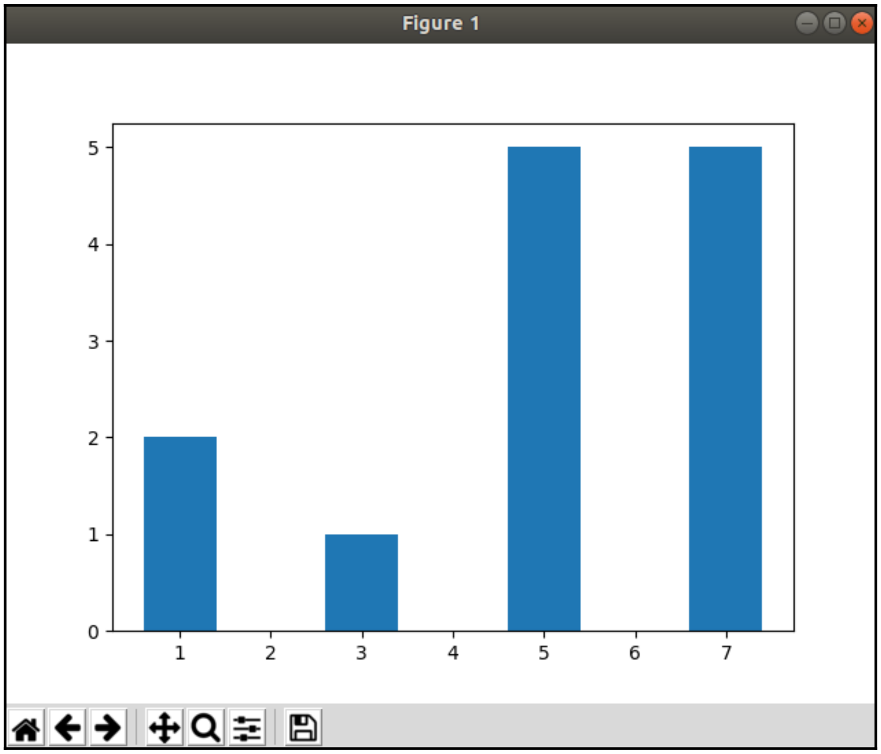
To generate a bar chart, instead of calling `plt.plot`, we have to call the `plt.bar` function. Now, keeping the same code as the linechart, we will make the changes as follows:

```
import matplotlib.pyplot as plt

x = [1, 5, 3, 7]
y = [2, 5, 1, 5]

plt.bar(x, y)
plt.show()
```

Save this file in the same `Matplotlib` folder using the name `barchart.py`. Now execute it using the following command:

```
$ python barchart.py
```

It will display output similar to the following:



Bar chart plotted using Matplotlib

We have now successfully created a bar chart using Matplotlib. We will now move on to creating a scatter chart.

If you want to add a heading in your chart, use the `plt.title` function. For example, `plt.title("My Bar Chart")`.

Stack charts

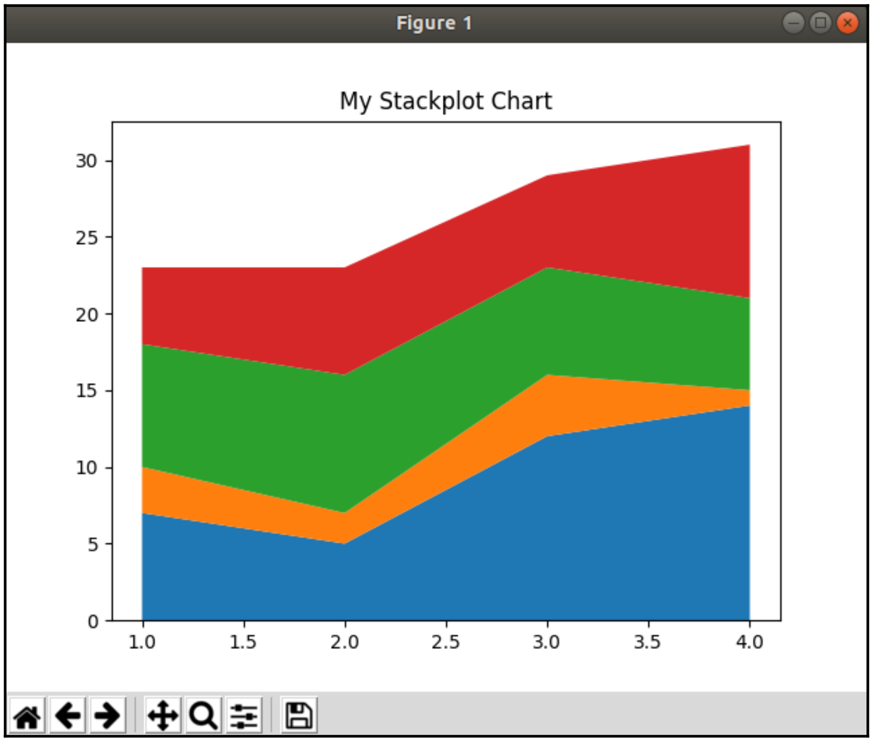
Let's now create a stack chart. A stack chart is used mainly to identify the minority and majority of the data it plots. Let's suppose we have defined `a` to be plotted on the *x* axis, which is nothing but a series of numbers depending on the number of elements in our data matrix. Then we have data of four different classes, `W`, `X`, `Y`, and `Z`, which we will plot on the *y*-axis, as shown in the following code:

```
import matplotlib.pyplot as plt

a = [1, 2, 3, 4]
w = [7, 5, 12, 14]
x = [3, 2, 4, 1]
y = [8, 9, 7, 6]
z = [5, 7, 6, 10]

plt.stackplot(a, w, x, y, z)
plt.title("My Stackplot Chart")
plt.show()
```

Save this file in the same `Matplotlib` folder using the name `stackchart.py`. Now execute it using the following command:

```
$ python stackchart.py
```

The following screenshot displays the output of this program:



Stack plot chart

We have successfully created a stack plot chart. We will now create a scatter chart.

Scatter charts

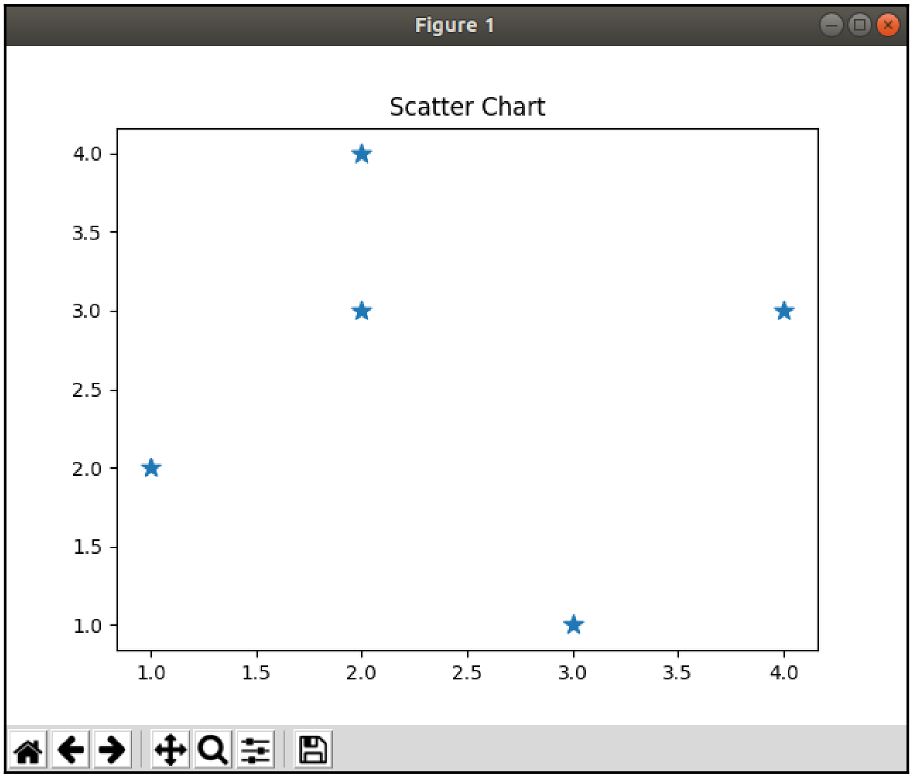
Let's now create scatter chart. In this type of chart, the data is represented on the chart with the help of a marker. We will be using the following code to create this type of graph:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 2, 4]
y = [2, 4, 1, 3, 3]

plt.scatter(x, y, marker="*", s=100)
plt.title("Scatter Chart")
plt.show()
```

Save this file in the same `Matplotlib` folder using the name `scatterchart.py`. Now execute it using the following command:

```
$ python scatterchart.py
```

The following screenshot displays the output of this program:



Scatter hart

In the code, we have defined the marker to be `*` with size `200`. It can be anything, such as `x` or `o`, with any size you want. Now that we have successfully created a scatter chart, we will now proceed to creating a pie chart.

Pie charts

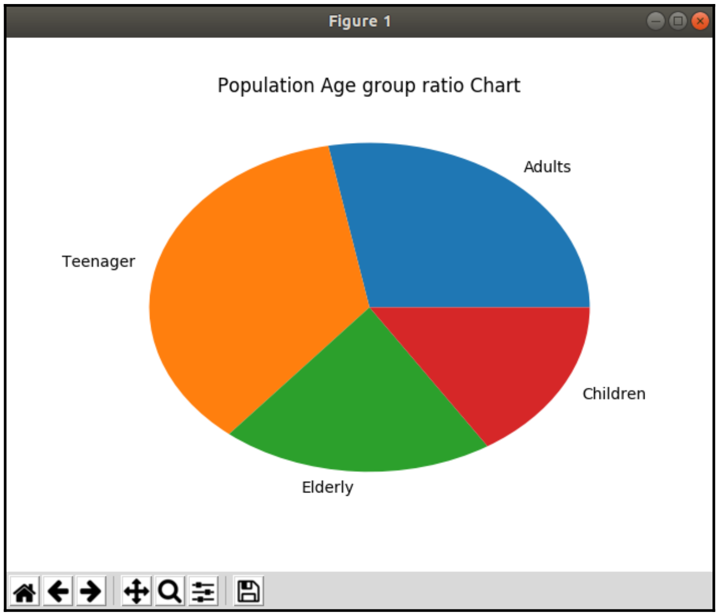
We will now create a pie chart. For this purpose, we have to define the labels for the age group and the data related to its ratio. In this example, we will plot data related to the age group ratio of the population. We will be using the following code to generate the pie chart:

```
import matplotlib.pyplot as plt

ratio = [7, 9, 5, 4]
agegroup = ["Adults", "Teenager", "Elderly", "Children"]

plt.pie(ratio, labels=agegroup)
plt.title("Population Age group ratio Chart")
plt.show()
```

Save this file in the same `Matplotlib` folder using the name `scatterchart.py`. Now execute it using the following command:

```
$ python scatterchart.py
```

The following screenshot displays the chart generated as a result of executing the program:



Population age-group ratio pie chart

We have now successfully created a pie chart. Let's move on to geographic projection sub charts.

Geographic projections

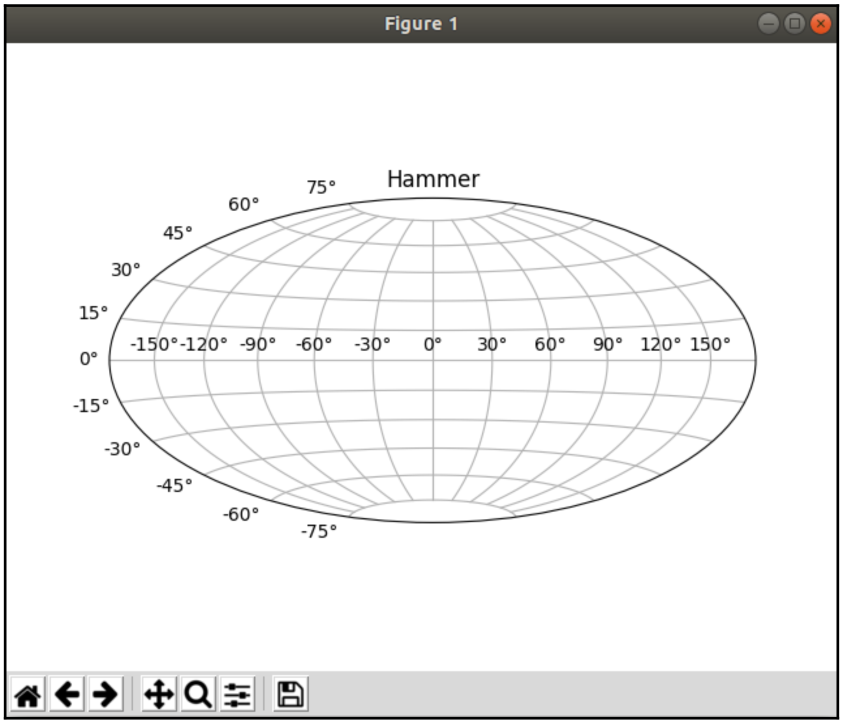
The `matplotlib` library has a geographic projection chart as a subplot. A subplot is a chart that you will draw along with an existing chart. Let's assume you have a scattered plot data that you want to highlight on the geographic projection, then it is the type of subplot you will need. The following is the code used to draw it geographic projection:

```
import matplotlib.pyplot as plt

plt.figure()
plt.subplot(111, projection="hammer")
plt.title("Hammer")
plt.grid(True)
plt.show()
```

Save this file in the same `Matplotlib` folder using the name `gp_hammer.py`. Now execute it using the following command:

```
$ python stackchart.py
```

The following screenshot displays the output of this command:



Geographic projection with the Hammer type

The following are the different projection types available to be used:

- `aitoff`
- `hammer`
- `lambert`
- `mollweide`

You can change it based on your requirement, as each one of them has its own variation. It is defined on the following line of code:

```
plt.subplot(111, projection="hammer")
```

With the help of the `matplotlib` library, we have created a line chart, bar chart, stack chart, scatter chart, pie chart, and a geographic projection as a subplot with very few lines of code. Let's now move on to the other data visualization tool, `D3.js`.

D3.js

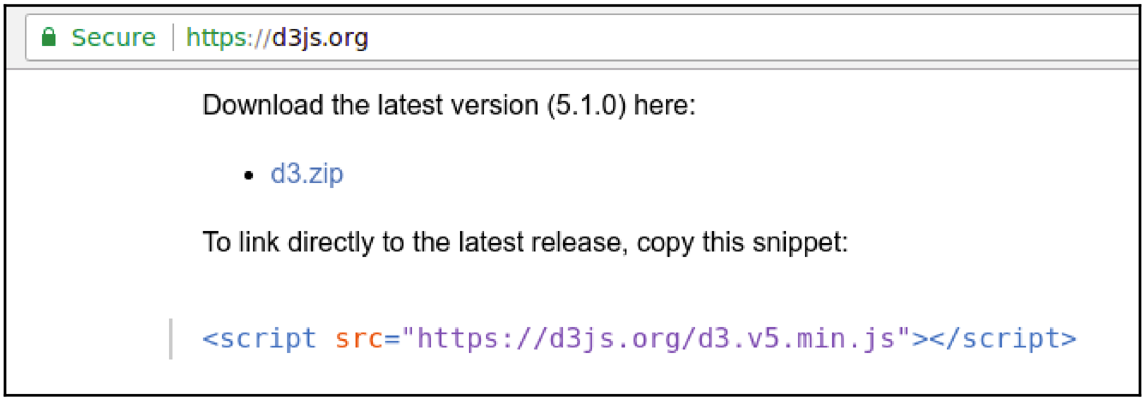
D3 stands for **Data-Driven Document**. It is a JavaScript library that uses **document object model** (**DOM**) elements to create different visual forms to represent data. The D3 framework combines with HTML, SVG, and CSS to create colorful as well as interactive objects on the web page, which helps the end user to understand data with more clarity.

D3 is a powerful framework. It first binds the data with DOM. Then it uses it to create different elements on the web page. The element can be anything. For example, if you want to represent your data in a tabular form, you can do it. If you want to display a pie chart using the same data, you can also do it. This architecture utilizes the most of the capability of modern browsers, leaving end users with colorful and interactive web pages without affecting the performance.

Let's now start with the installation of the `D3.js` library.

Installation

D3.js is a JavaScript library, which means you have to include the D3.js-dependent JavaScript file into your web page where you want to use it. There are two ways to do this and the following screenshot illustrates how to do it from `D3js.org`, the official website of the framework:



Secure | https://d3js.org

Download the latest version (5.1.0) here:

- d3.zip

To link directly to the latest release, copy this snippet:

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

Installation option for the D3.js library

The first option is to download the required files, manually include them in you web project, and then link the script file in you web page where you want to use the functionality of the D3.js library. The second option is to leave the handling of script files and just copy the code from here and paste it on to your web page, where you want to utilize it. The decision is up to you.

Practical example

It's time to dive in to the code. We will create a bar chart using the D3.js library. For this purpose, we will be creating three files. The first one is an HTML file, which will be the main file that holds the basic information and links to other files. The second one is a CSS file, which contains the styling of the chart, and the last one is a JavaScript file, which will contain the code to create a bar chart. Let's start with the HTML file.

The HTML file is similar to any other HTML file. It will contain the head section and the body section. In the head section, we will link the D3.js script file and link to our CSS file. The body contains the link to the JavaScript file that contains the code to create a bar chart. The complete HTML files will look as follows:

```
<html>
<head>

<title>Barchart Example</title>
<script src="https://d3js.org/d3.v5.min.js"></script>
<link rel = "stylesheet" type = "text/css" href="barchart.css" />

</head>

<body>

<h1>Barchart</h1>
<script src="barchart.js"></script>

</body>
</html>
```

Save this code as `barchart.html` in a folder. Remember to create the remaining JavaScript and CSS files in the same folder where you put this HTML file; otherwise, point the links of these file correctly if you decide to save them in a different folder.

We will now create a JavaScript file. It contains the data the we will be plotting in the form of a bar chart. The following code represents the sample data:

```
var chartdata= [120,100, 20, 40, 70, 150, 10, 30, 50, 250];
```

Now that we have our data, it's time to create the `D3.js` element to utilize it to create a bar chart. The following code will be used for this purpose:

```
d3.select("body").selectAll("div")
  .data(chartdata)
  .enter()
  .append("div")
  .attr("class","bar")
  .style("height", function(d) {
    return d + "px";
  });
```

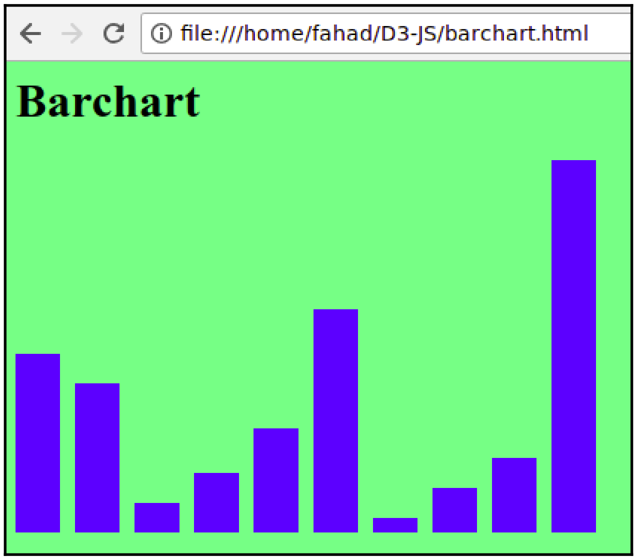
Save both the codes in a file named `barchart.js` and place it in the same folder as the HTML file. Now the last one is the CSS file, which will contain the styling of the bars in the charts. The code of CSS is as follows:

```
body {
  background-color: #8FE985;
}
div.bar {
  display: inline-block;
  width:30px;
  background-color:blue;
  margin-right:10px;
}
```

Save both the codes in a file named `barchart.css` and place it in the same folder with the HTML and JavaScript files.

Output

Now that we have completed the code, it's time to test it by opening `barchart.html` in your favorite browser. The following screenshot displays the output of the program:



Bar chart using D3.js

The preceding confirms the creation of the bar chart was successful. You can play around with different stylings by editing the CSS file, modifying the data to match your own needs, or completely changing the web page to best fit your needs. The limit is only based on your imagination.

There is a very extensive list of examples of different types of visual forms that can be created using D3.js, available at `https://github.com/d3/d3/wiki/Gallery`. It's worth checking out.

Summary

Sometimes, data doesn't seem worthy if not presented in a way it should be. In this chapter, we have gone through two open source libraries to format your data in a more presentable way. The first library we explored was Matplotlib, which is based on Python. The object of this library is to create different types of charts with minimal coding. We started with the installation of Matplotlib and then moved on to some practical examples, where we created different types of charts. These charts included a line chart, bar chart, pie chart, and a scatter chart.

Then we extended our knowledge to the second library, named D3.js. It is based on JavaScript and is used to create different colorful and interactive data representations. It utilizes the power of DOM to increase performance and reuses the data to create different visual elements. We started with an introduction to D3.js. Then we explored the two different options of using the core library framework in our web project. We then proceeded further with a practical example, in which we created a bar chart on the web. That example consisted of creating three different files: HTML, CSS, and JavaScript. After writing the code, we placed all the files in the same folder and opened the web page, which in turn contains the desired output in the form of a bar chart.

In the next chapter, we will introduce algorithmic trading, the who, what, why, when, and how of algorithmic trading, as well as different trends and trading strategies for it.

18
Financial Trading System

Each and every industry is now using computer-related technologies—from simple office work to advanced robotic work—ranging from the postal service, fashion, and shopping, to medical applications. Even the technology itself is being used to create new technology. More and more industries are now looking into different ways to effectively use advanced technology and **artificial intelligence** (**AI**) in their day-to-day work to take it to the next level.

The financial industry is no different than any other industry. The people in it are also working on different approaches to make more money, using the help of computer systems and artificial intelligence, especially in the trading market sector. The essence of their work is time, and for them, time is money, literally. Because sometimes, within a fraction of seconds, they make a lot of money, and sometimes they can lose a significant amount if they don't act within the available time slot. In this chapter, we will be covering the following topics, which are directly related to the financial market:

- Algorithmic trading
- Benefits of algorithmic trading
- Different strategies for algorithmic trading
- Building our own Expert Advisor
- Backtesting historical data

Let's start our discussion with our first topic and question: What is algorithmic trading?

What is algorithmic trading?

In the financial world, when you buy or sell anything, it is called **trade**. It can be real estate, currency, shares, or goods. This normally happens between two parities. A representative of each party talks to the other and makes the trade happen. The medium they used in earlier days was paper-based, and in time it was replaced by computer and mobile applications. However, the end user is still a human being who makes sure that everything is in order.

Now as the technology is advancing and people are looking to automate most of their work, the same thing applies to the financial market. In the financial market, we talk specifically about shares and the Forex market, it is the most liquid market. The purchase and sale happens there so fast that every second counts. Sometimes, if you flinch in making a decision, you may lose big money.

The people in that market are now trying to come up with an automated way of doing trades, leaving a human as a supervisor. This automation works based on a set of predefined rules to govern them. This leads to the buying and selling of financial instruments in a matter of milliseconds, as they don't flinch, unlike humans, when making decisions. This automated way of doing trade is called **algorithmic trading**. It is also referred to as algo-trade, black-box trading, and automated trading.

These algorithms are built using the experience of the traders, the type of factors they monitor, the opportunities they look for, the calculations they perform before making a trade that eventually leads to a successful trade. The stronger your algorithm, the better the chances that you are going to make money.

Let's now discuss some of the key benefits of using algorithmic trading.

Benefits of algorithmic trading

These are some of the key benefits of using algorithmic trading, if you have a good and strong algorithm in place:

- There is always a chance of human error in any kind of work, for various reasons. Using algorithmic trading can avoid this as it acts based on predefined rules.
- You can watch multiple markets at the same time and trade on them simultaneously, which is impossible for a human person to do.
- It will place the trade in a fraction of a second once the criteria or the rule matches the market condition.

- You can apply multiple sets of instructions to automated, which it will watch and act accordingly.
- It reduces the risk factor that may be effected by physiological and emotional human factors.
- Due to its instant, human-free behavior, it may get the best possible price for the trade.
- This automation can place a trade on the desired item accurately and instantly, which leads to an increase in revenue.

These are some of the key benefits of using algorithmic trading. We will now discuss the involvement of big data in the financial sector.

Big data in the financial market

At this stage, we already know about big data and its characteristics. It helps companies to make more robust, intelligent, and powerful decisions using the magic of data. The two main characteristics that the financial market possess from the big data world are volume and velocity. The data in the financial market is too huge and generating in a very high velocity, that if you store and analyze that data, it will help to generate good revenue out of it.

Now, when you develop an algorithm to be used in your trading, will you apply it directly on live trading? No, because if it contains any logical or technical error, you may lose everything. Nor is it reliable to use on dummy data, because it may not project the actual market accurately. In the actual market, everything is not black and white. There are many cases where the market was performing normally, but due to a sudden crash, many investors went bankrupt.

To avoid these types of cases while utilizing the full potential of technology and AI, the market data is getting stored and analyzed to identify different patterns and build different strategies on top of it. It is available to everyone, including researchers and investors, to play around with. This helps you to apply your algorithm and strategies on the historical data, to test whether your algorithm or strategy will help you generate revenue or lose money.

Let's now move on to discuss some strategies that traders used to place trades in the financial market.

Algorithmic trading strategies

Trading strategies are employed to identify potential opportunities to make money. In a traditional system, they require an experienced trader to do this job. In algorithmic trading, people are trying to use the expertise of an experienced trader to make the algorithm robust and powerful. Few are successful because the market is not predictable or works based on mathematical models. Still. people are making money using this approach. Let's now discuss some of these strategies that help in identifying a potential trade:

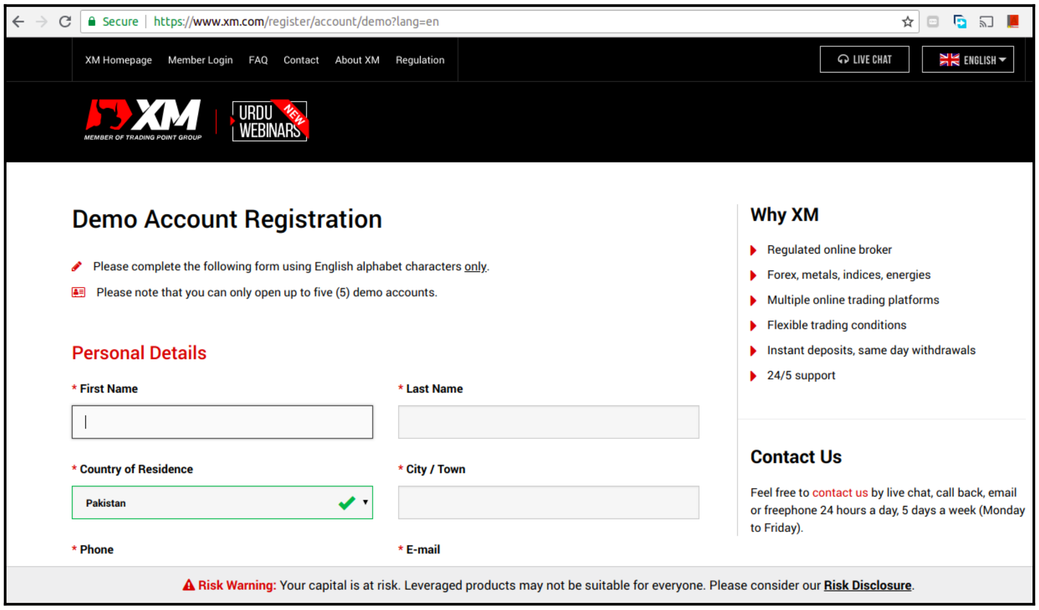
- **Trend-following strategy**: This is the most common type of strategy that most traders use, whether it is to perform an actual trade or just to do the analysis. This strategy doesn't involve making any sort of prediction and is purely based on the mathematical model. It will trigger a trade when the price follows a predefined trend. If the current price is going higher than the moving average and it is less than the last updated price, then it's time to sell. Similarly, if the current price is greater than the last price and the current price is less than the moving average, then it's time to buy.
- **Arbitrage opportunities**: In this type of strategy, a trader buys an instrument from one market and at the same time, trades to sell it for a higher price on another market. This strategy can be applied to different markets, such as Forex and stocks. If it is applied properly to identify and calculate the differential price, it can provide many profitable opportunities.
- **Mean reversion**: This strategy works based on the idea that everything comes back to its initial stage or at least close to it. Using this approach, traders calculate the base value of the instrument and whenever the price changes to break in or out of that range, they go for the trade.
- **Volume-weighted average price**: In this type of strategy, a large order will be broken down into smaller chunks based on the historical volume price within a given time frame. This type of strategy usually applies to pension and mutual funds
- **Time-weighted average price**: In this type of strategy, a large order will be broken down into smaller chunks based on the evenly-divided time slots between a start and an end time. The trade only happens when the price gets close to the average price between the start and end time. It will minimize the market impact on the trade.

- **Percentage of volume**: In this strategy, the volume distribution and the order quantity over a period of time is determined, using the target percentage of the volume along with the volume-forecast calculated updates from the market. It allows us to participate based on the volume in the custom defined rate. It is sort of issue partial orders to the market until the a complete large order takes place. It is also referred to as a **step strategy**.

Now that we have learned some of the strategies used by traders in algorithmic trading, it's time to create our own algo-trader, based on one of the strategies that we have just discussed.

Building an Expert Advisor

We will now go through the steps of creating our own Expert Advisor/trading bot and perform algorithmic trading. For this purpose, will be register a demo account in one of the leading foreign-exchange brokers, XM. Visit their website, `https://www.xm.com/`, and click on the **OPEN DEMO ACCOUNT** link, or directly visit `https://www.xm.com/register/account/demo?lang=en`. It will open up the registration page, as shown in the following screenshot:


Secure | https://www.xm.com/register/account/demo?lang=en
XM Homepage
Member Login
FAQ
Contact
About XM
Regulation
LIVE CHAT
ENGLISH
XM
MEMBER OF TRADING POINT GROUP
URDU WEBINARS
NEW

Demo Account Registration

Please complete the following form using English alphabet characters only.

Please note that you can only open up to five (5) demo accounts.

Personal Details

* First Name	* Last Name
* Country of Residence: Pakistan	* City / Town
* Phone	* E-mail

Why XM

- Regulated online broker
- Forex, metals, indices, energies
- Multiple online trading platforms
- Flexible trading conditions
- Instant deposits, same day withdrawals
- 24/5 support

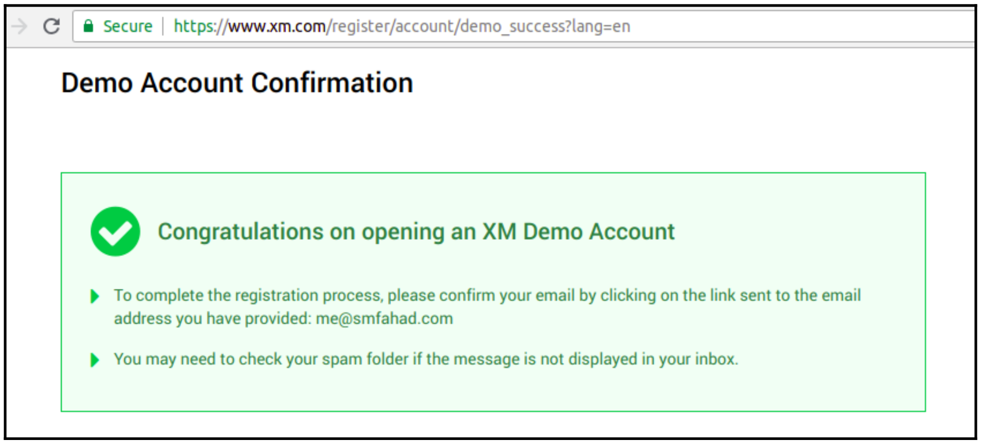
Contact Us

Feel free to contact us by live chat, call back, email or freephone 24 hours a day, 5 days a week (Monday to Friday).

Risk Warning: Your capital is at risk. Leveraged products may not be suitable for everyone. Please consider our Risk Disclosure.

XM demo account registration page

It will ask you to provide some basic information, such as name, email, phone number, account password, country, and city. It will also show you some other trading settings, such as trading platform, account type, account base currency, leverage, and investment amount. For now, we will not touch these settings and keep it on the default settings. Once you fill out the form, click on the `Open a Demo Account` button at the end of the page. After submitting the form, you will see a success message similar to the following screenshot:



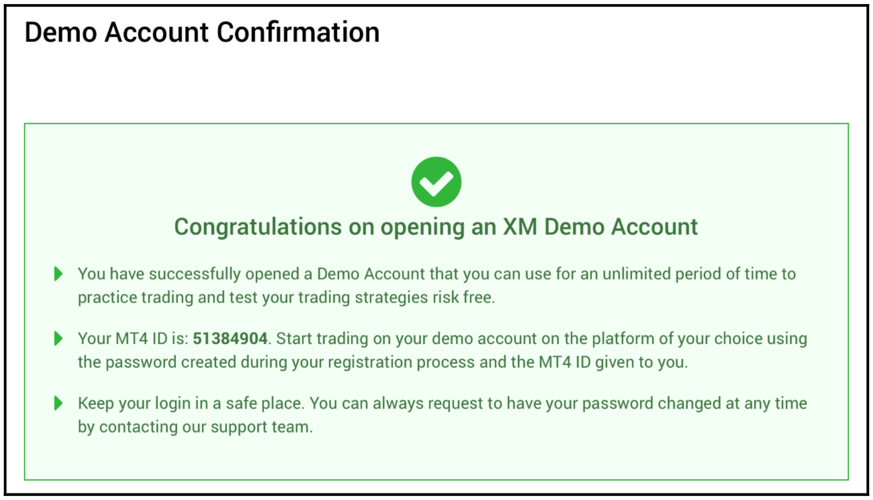
Demo Account Confirmation

Congratulations on opening an XM Demo Account

- To complete the registration process, please confirm your email by clicking on the link sent to the email address you have provided: me@smfahad.com
- You may need to check your spam folder if the message is not displayed in your inbox.

XM success message of opening a demo account

The next step is to confirm your email address. XM sends an email to your provided email address with a confirmation link. Just click on the **Confirm Email Address** button to proceed to the next step. The new page that will pop up will contain the MT4 ID that is required to log in to the trading application for the Forex market, as shown in the following screenshot:



Demo Account Confirmation

Congratulations on opening an XM Demo Account

- You have successfully opened a Demo Account that you can use for an unlimited period of time to practice trading and test your trading strategies risk free.
- Your MT4 ID is: **51384904**. Start trading on your demo account on the platform of your choice using the password created during your registration process and the MT4 ID given to you.
- Keep your login in a safe place. You can always request to have your password changed at any time by contacting our support team.

XM confirmation page with new MT4 ID

Now that we have created and confirmed our account from XM, it's now time to start working on the actual Forex trading station, but before that, let's have a brief introduction to the trading platform that we are going to use.

MetaTrader

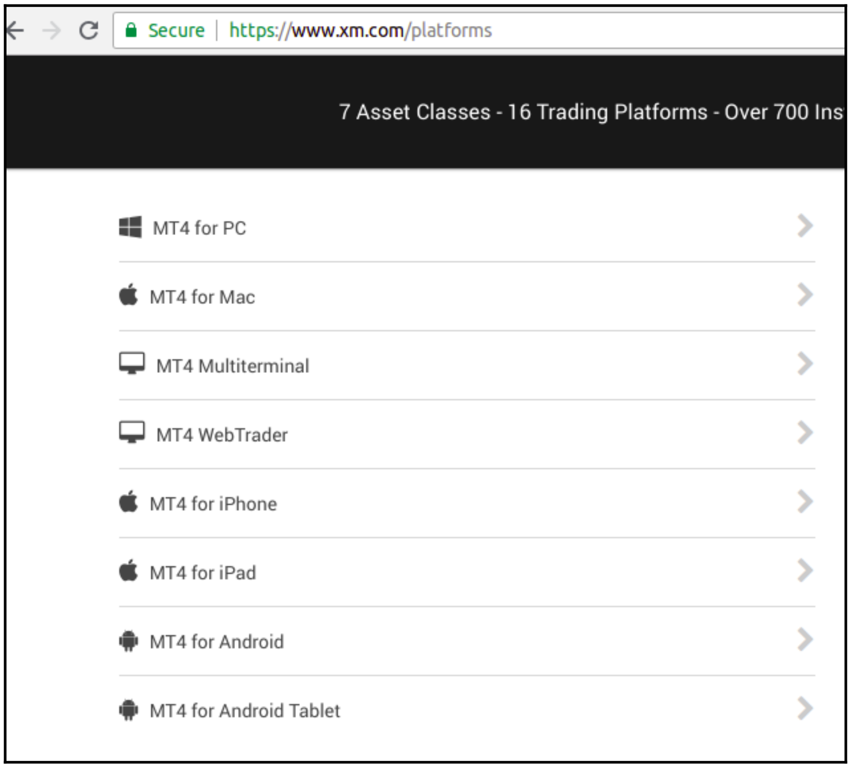
There are many trading stations available in the market. In most cases, all the brokers are using a common trading station, customized per their requirements. The most common among brokers is MetaTrader, which is commonly known as the MT trading station. It is developed by the MetaQuotes software company.

It has two components: the server application and the client application. The client version will communicate with the broker server version to verify the client account and its related customized settings. The first version of MetaTrader was released in 2002, and from time to time, they released different version of MetaTrader. Version 4 was widely adopted by many brokers, as it was well enhanced and have the functionality to write its own code scripts to automate the trading process. It can just simulate the trading for analysis purpose and can also perform the actual trade using these automated algorithms. The company also released the next version of MetaTrader, but many brokers still use MT4, as the new version has few changes.

Let's now move on to download MetaTrader and set it up.

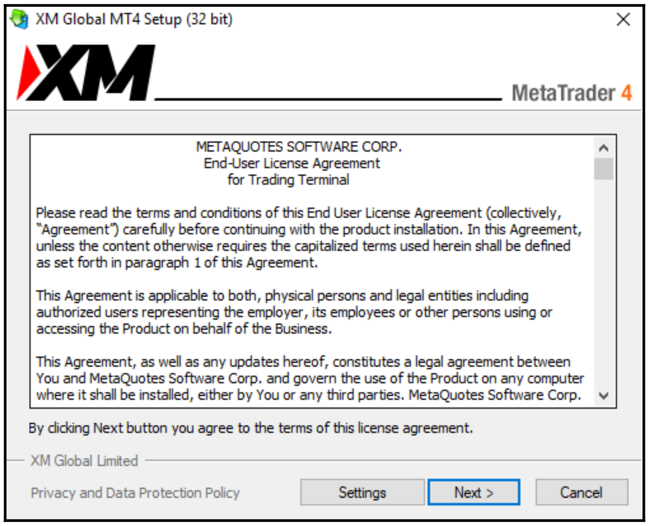
Downloading and setting up MetaTrader

We will now download the MetaTrader application by pointing our browser to `https://www.xm.com/platforms`. It will open a page that will list all the platforms that MetaTrader supports. The following figure shows the list of platforms it supports:



Platforms supported by MetaTrader 4 by XM

In our case, we will be using **MT4 for PC**. Click on it and, on the next page, click on the **Download** button to start downloading it. Once you click on it, it will start downloading a file that is approximately `1.2` MB in size. After the download finishes, double-click on the file to begin the installation process. It will open up the application window, as shown in the following screenshot:



XM Global MT4 Setup (32 bit)

METAQUOTES SOFTWARE CORP.
End-User License Agreement
for Trading Terminal

Please read the terms and conditions of this End User License Agreement (collectively, "Agreement") carefully before continuing with the product installation. In this Agreement, unless the content otherwise requires the capitalized terms used herein shall be defined as set forth in paragraph 1 of this Agreement.

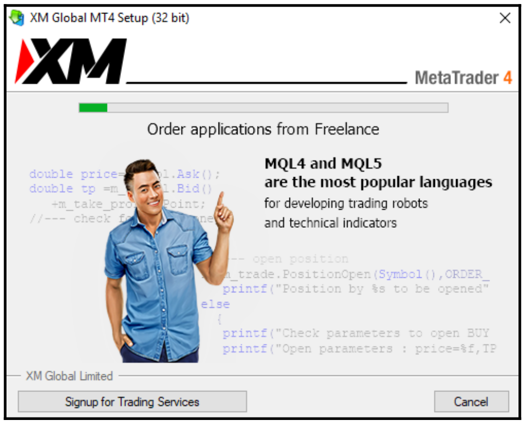
This Agreement is applicable to both, physical persons and legal entities including authorized users representing the employer, its employees or other persons using or accessing the Product on behalf of the Business.

This Agreement, as well as any updates hereof, constitutes a legal agreement between You and MetaQuotes Software Corp. and govern the use of the Product on any computer where it shall be installed, either by You or any third parties. MetaQuotes Software Corp.

By clicking Next button you agree to the terms of this license agreement.

XM MetaTrader4 installation

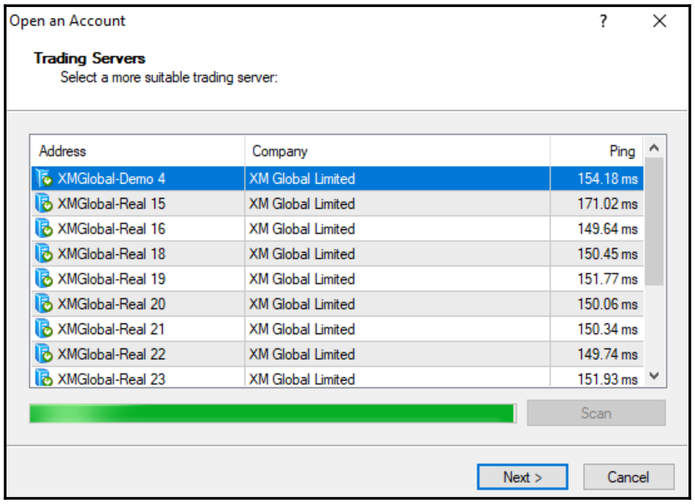
Click on the **Next** button to continue the installation process. It will now download the actual application, which will take around four to five minutes, depending on your internet connection speed. Take a look at this screenshot:



XM Global MT4 Setup (32 bit)
MetaTrader 4
Order applications from Freelance
MQL4 and MQL5
are the most popular languages
for developing trading robots
and technical indicators
XM Global Limited
Signup for Trading Services
Cancel

XM MetaTrader 4 installation

Once it finishes downloading the required files, it will silently install them and open the application:



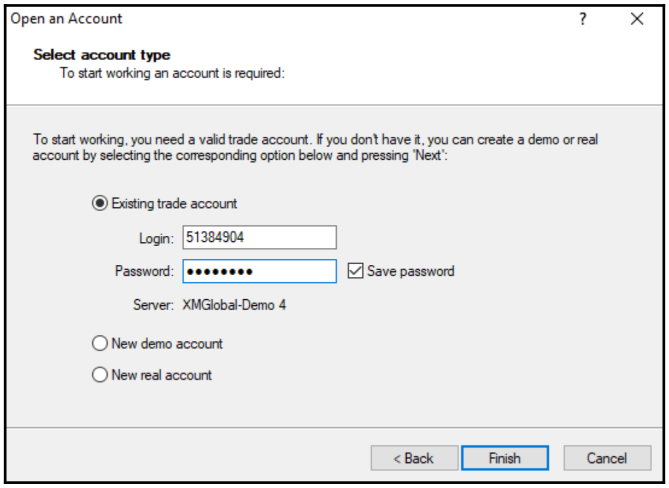
Open an Account

Trading Servers
Select a more suitable trading server:

Address	Company	Ping
XMGlobal-Demo 4	XM Global Limited	154.18 ms
XMGlobal-Real 15	XM Global Limited	171.02 ms
XMGlobal-Real 16	XM Global Limited	149.64 ms
XMGlobal-Real 18	XM Global Limited	150.45 ms
XMGlobal-Real 19	XM Global Limited	151.77 ms
XMGlobal-Real 20	XM Global Limited	150.06 ms
XMGlobal-Real 21	XM Global Limited	150.34 ms
XMGlobal-Real 22	XM Global Limited	149.74 ms
XMGlobal-Real 23	XM Global Limited	151.93 ms

XM MetaTrader 4 application first run

If you are viewing this screen, then it means that MetaTrader4 has been successfully installed in your computer. Now select `XMGlobal-Demo 4` from this screen, and click next to step the login for MT4 trading stations. It will open the following screenshot:



Open an Account

Select account type
To start working an account is required:

To start working, you need a valid trade account. If you don't have it, you can create a demo or real account by selecting the corresponding option below and pressing 'Next':

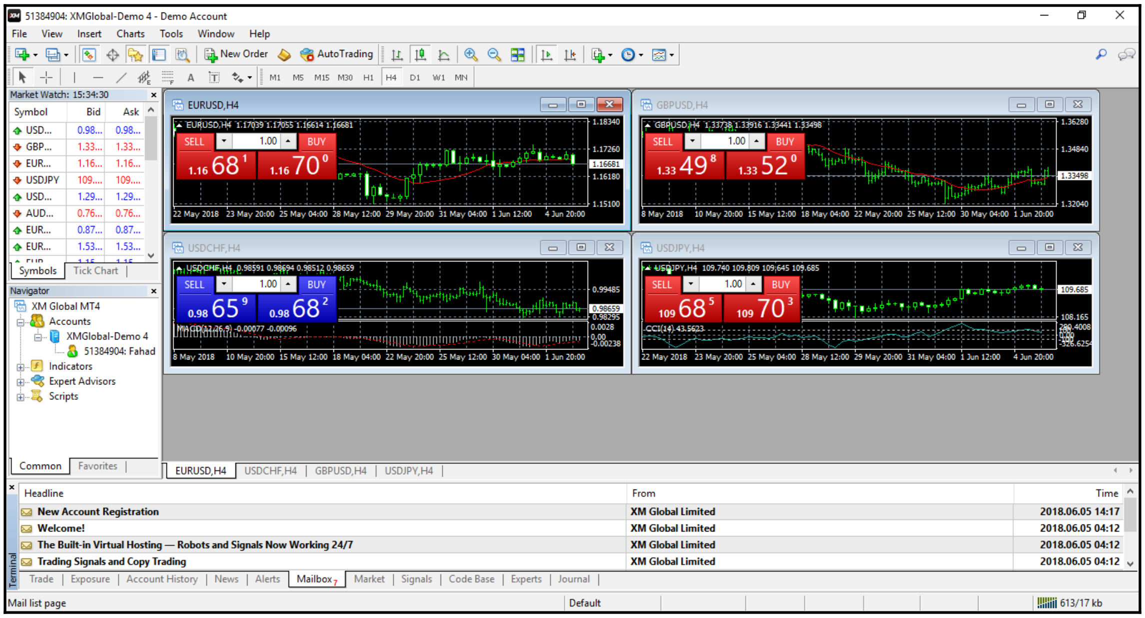
● Existing trade account

Login: 51384904
Password: •••••••• ☑ Save password
Server: XMGlobal-Demo 4

○ New demo account
○ New real account

XM MetaTrader4 login

We have already created our demo account from XM and we have received the MT4 ID. We will put this ID in the `Login` textbox along with the password we used to sign up for the demo account. Fill in these details and click on `Finish` to log in to MT4. You will see a screen similar to the following screenshot:


51384904: XMGlobal-Demo 4 - Demo Account
File View Insert Charts Tools Window Help
New Order
AutoTrading
Market Watch: 15:34:30
Navigator
XM Global MT4
Accounts
XMGlobal-Demo 4
51384904: Fahad
Indicators
Expert Advisors
Scripts
Common Favorites
EURUSD,H4 USDCHF,H4 GBPUSD,H4 USDJPY,H4
Trade Exposure Account History News Alerts Mailbox Market Signals Code Base Experts Journal
Mail list page
Default
613/17 kb

Headline	From	Time
New Account Registration	XM Global Limited	2018.06.05 14:17
Welcome!	XM Global Limited	2018.06.05 04:12
The Built-in Virtual Hosting — Robots and Signals Now Working 24/7	XM Global Limited	2018.06.05 04:12
Trading Signals and Copy Trading	XM Global Limited	2018.06.05 04:12

MetaTrader 4 welcome screen

If you have reached to this screen, then congratulations, you are all set up to go to the next step.

MetaQuotes language

We will now create an automated trading bot in MetaTrader. MetaTrader uses the MQL programming language to create automated analysis and trading bots. If you have experience in any of the modern programming, that's a big plus, otherwise it will be little difficult to get you started. I will keep this exercise simple and understandable.

The MetaTrader application has a built-in Meta Editor that will let you write codes and execute them. The following are the three types of program that you can code in MetaTrader:

- **Scripts**: They are created to execute one time on a given chart, it can be to indicate or draw something or to do a calculation.
- **Indicators**: It draws indicators of the live charts based on the algorithm and strategy you apply. Some of the traders prefer the way to indicate the potential trade based on the calculation and strategy and then they trade manually. It can have many other purposes, such as to analyze the market in real time.
- **Expert Advisors**: It is created to apply the algorithm and strategy on the live data and, based on the output of the strategy, perform the actual buy and sell, and make the trade happens without any third-party involvement. Once you execute them, it will make the decision based on the programmed strategy and algorithm and fly solo in the live market.

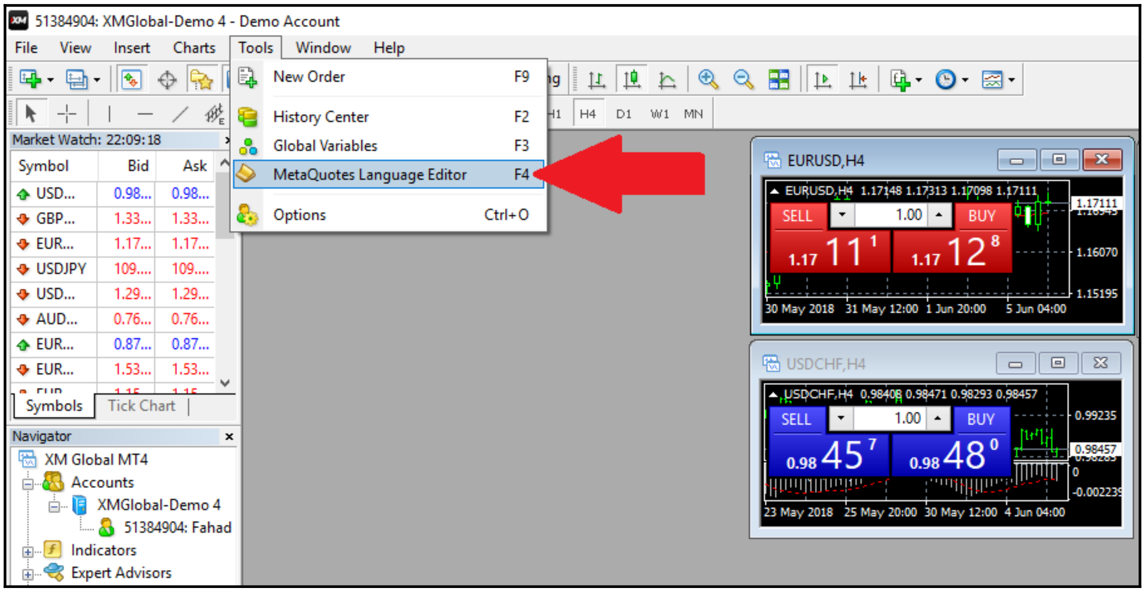
Let's now move forward and understand the objective of creating this trading bot.

Trading bot objective

The object of creating this trading bot is to buy and sell trades in the live market based on the strategy we programmed. In the preceding section, we learned some trading strategies, out of which we will be using the moving-average strategy. This trading bot will buy and sell the currency once the moving-average strategy triggers it.

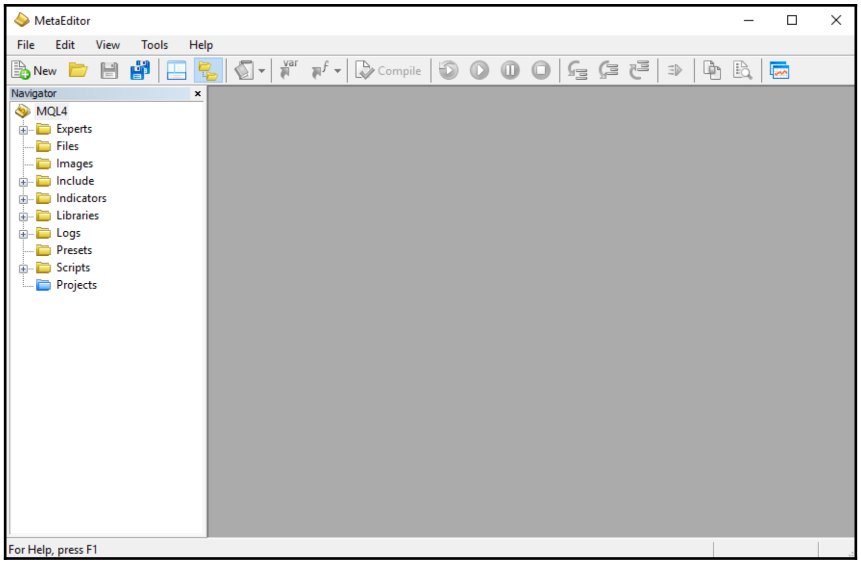
Practical

Now that we have set up our environment and have our clear objective of created an algorithm trading bot, we are all set to move to the next step. We will now launch Meta Editor to start building our own trading bot. From the GUI, click on the `Tool` menu and then click on `MetaQuotes Language Editor`, the shortcut for it is `F4`:



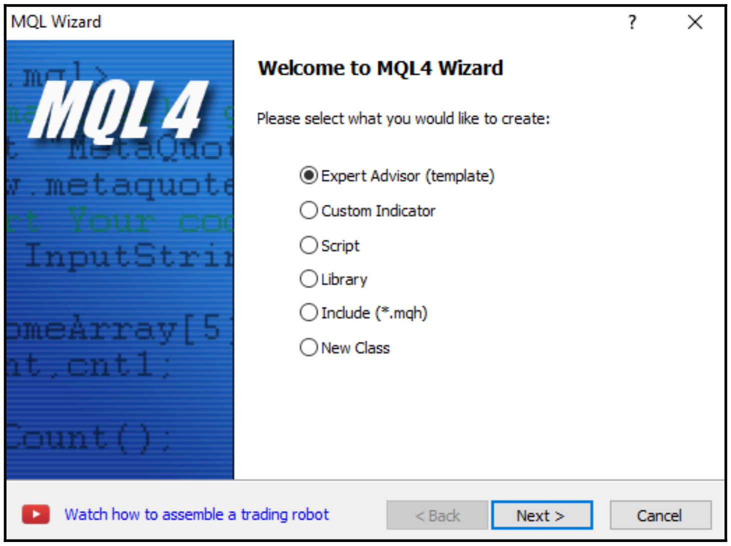
Starting up the MetaQuotes language editor

Once you click on it, it will open Meta Editor. This is the main screen that we will be working on. It looks similar to the following screenshot:



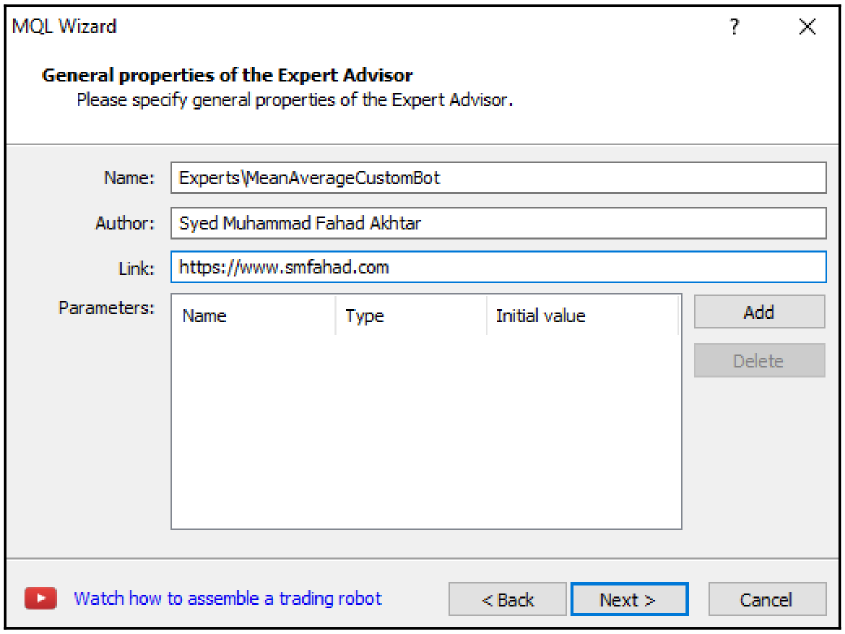
MetaEditor welcome screen

To create our first Trading bot, click on the **File** menu from the top-left side, and click on **New** to start creating a bot. Once you click on it, it will display a window similar to the following screenshot:



Select the type of program to create in MetaTrader

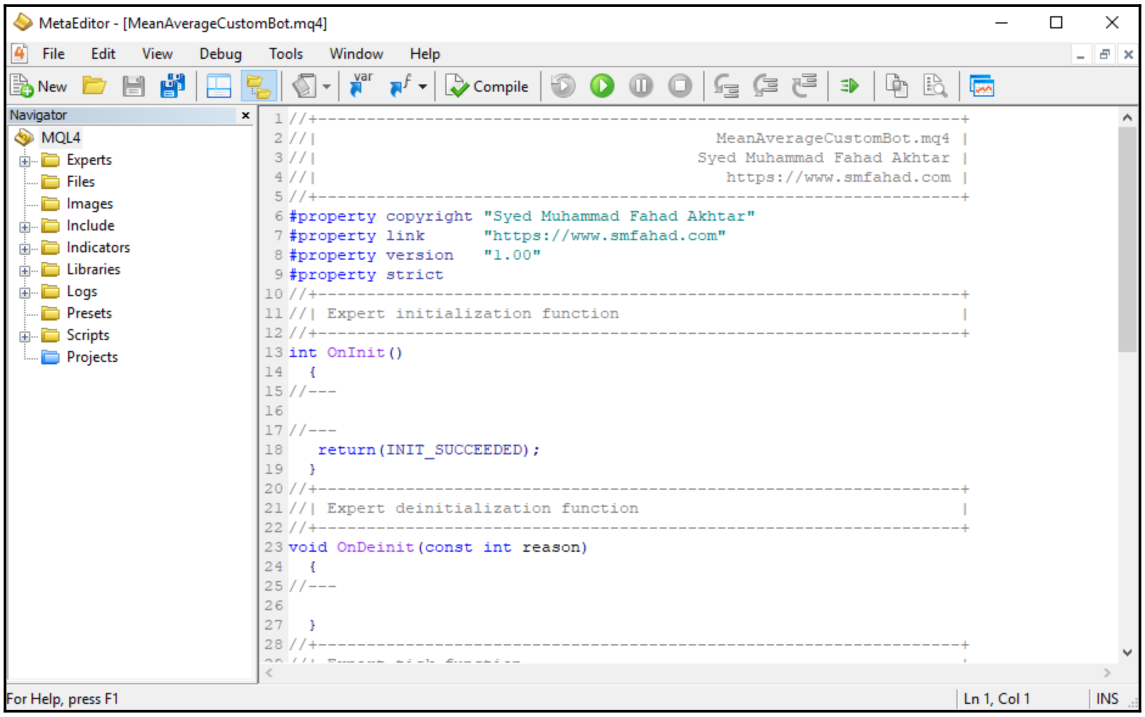
We have already discussed the difference between these options in the *MetaTrader* section of this chapter. For now, we will select **Expert Advisor (template)** and click **Next**. It will then display a form to input some basic information, such as the **Name** of the advisor, the **Author**, and the **Link** to a reference website. Refer to the following screenshot:


MQL Wizard
?
General properties of the Expert Advisor
Please specify general properties of the Expert Advisor.
Parameters:
Name
Type
Initial value
Add
Delete
Watch how to assemble a trading robot
< Back
Next >
Cancel

Name: Experts\MeanAverageCustomBot
Author: Syed Muhammad Fahad Akhtar
Link: https://www.smfahad.com

Input parameters in MetaTrader

Once you fill out the details, click on the Next button. It will display some more advanced options to select. But for now, we will skip those. Keep clicking the **Next** button until you reach to the final step, and click **Finish**. Once you click finish, it will display a code window, as shown in the following screenshot:



```
//+------------------------------------------------------------------+
//|                                         MeanAverageCustomBot.mq4 |
//|                                       Syed Muhammad Fahad Akhtar |
//|                                           https://www.smfahad.com |
//+------------------------------------------------------------------+
#property copyright "Syed Muhammad Fahad Akhtar"
#property link      "https://www.smfahad.com"
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---

  }
//+------------------------------------------------------------------+
```

Code editor window of Meta Editor

The code that will come by default contains the structure of the following functions to get you started:

- `OnInit()`: This function will be called only once when you start the program. It may be used to initialize the variable or to called some other onetime `call` function.
- `OnDeinit()`: This function will be called only once when you stop the program. It may be used to deinitialize any globally-defined variable.
- `OnTick()`: This function will be called every time the price on the given chart updates.

The main function that we will work on for now is `OnTick()`.

Trading pattern – moving average

We will now write our code to calculate the moving average. Meta Editor provides a built-in `iMA` function to calculate the moving average. It requires the following input parameters, as per the definition provided:

```
double iMA(
 string symbol, // symbol
 int timeframe, // timeframe
 int ma_period, // MA averaging period
 int ma_shift, // MA shift
 int ma_method, // averaging method
 int applied_price, // applied price
 int shift // shift
 );
```

For now, we will keep it simple and use the following parameter to start calculating the moving average. Write this code inside the `OnTick()` function to calculate the values on each price update:

```
double movAverage = iMA (NULL, 0, 15, 0, MODE_SMA, PRICE_MEDIAN, 0);
```

To verify that it is working as expected, use the following code to give a popup message with the calculated value inside the `OnTick()` function:

```
Alert("Moving Average Value : " + movAverage);
```

Decision time: buy or sell

We will now calculate the price variation to decide whether to initiate a trade for a buy or sell. For this purpose, we will first identify the value of the current price and the last updated price by writing the following code:

```
double currentPrice = Open[0];
double lastPrice = Open[1];
```

`Open` is a global array provided by MetaTrader to identify the change in price on the given chart. If it is on index `0`, that means the current value, if it is on index `1`, that means the last updated values, if it is on index `2`, that last to last updated value, and so on. The following code will be triggered once the selling condition meets the criteria for a trade:

```
if ( (currentPrice < lastPrice)
     && (currentPrice > movAverage) ) {
    // Place order to Sell
}
```

Similarly, the following condition will be triggered once the buying condition meets the criteria:

```
if ( (currentPrice > lastPrice)
    && (currentPrice < movAverage) ) {
    // Place order to Buy
}
```

Now we have reached the point where we will perform the actual trade. The function that will be used to place the order is `OrderSend`, which has the following parameters:

```
int  OrderSend(
   string   symbol,              // symbol
   int      cmd,                 // operation
   double   volume,              // volume
   double   price,               // price
   int      slippage,            // slippage
   double   stoploss,            // stop loss
   double   takeprofit,          // take profit
   string   comment=NULL,        // comment
   int      magic=0,             // magic number
   datetime expiration=0,        // pending order expiration
   color    arrow_color=clrNONE  // color
   );
```

To keep it simple, we will be calling the `OrderSend` function with the following parameters in order to place the order for `Sell`:

```
OrderSend(NULL, OP_SELL, 1, currentPrice, 0, NULL, NULL, NULL, 0, 0, NULL)
```

Similarly, to place the order to `Buy`, we will be calling the `OrderSend` function with the following parameters:

```
OrderSend(NULL, OP_BUY, 1, currentPrice, 0, NULL, NULL, NULL, 0, 0, NULL)
```

Complete program

The complete code of our moving-average Expert Advisor is as follows:

```
//+------------------------------------------------------------+
//| MeanAverageCustomBot.mq4 |
//| Syed Muhammad Fahad Akhtar |
//| https://www.smfahad.com |
//+------------------------------------------------------------+
#property copyright "Syed Muhammad Fahad Akhtar"
#property link "https://www.smfahad.com"
```

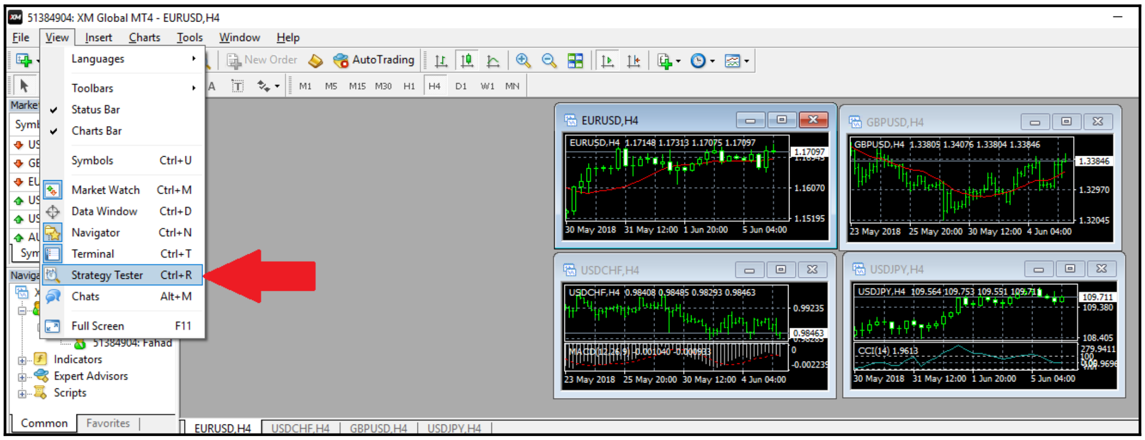
```
#property version "1.00"
#property strict
//+------------------------------------------------------------+
//| Expert initialization function |
//+------------------------------------------------------------+
int OnInit()
  {
//---
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------+
//| Expert deinitialization function |
//+------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
  }
//+------------------------------------------------------------+
//| Expert tick function |
//+------------------------------------------------------------+
void OnTick()
  {
      double movAverage = iMA (NULL, 0, 15, 0, MODE_SMA, PRICE_MEDIAN, 0);

      double currentPrice = Open[0];
      double lastPrice = Open[1];
      if ( (currentPrice < lastPrice)
           && (currentPrice > movAverage) ) {
        OrderSend(NULL, OP_SELL, 1, currentPrice, 0, NULL, NULL, NULL, 0,
0, NULL)
      }
      if ( (currentPrice > lastPrice)
          && (currentPrice < movAverage) ) {
        OrderSend(NULL, OP_BUY, 1, currentPrice, 0, NULL, NULL, NULL, 0, 0,
NULL)
      }
  }
//+-----------------------------------------------------------+
```

We will now perform backtesting on historical data to check how well our trading bot performed with the actual data.

Disclaimer: Don't try to run your Expert Advisor, including the one we just created, on the live market unless it has been thoroughly tested on historical data using backtesting.

Backtesting in MetaTrader 4

MetaTrader provides the option to backtest your strategy on historical data, so that you will come to know how your algorithm and strategy work on the actual data. The date range depends for the algorithm on you. Let's now run our trading bot on the back dates using the Strategy tester tool, which can be triggered from the View menu in the top-left, and then click on Strategy Tester to open it, as shown in this screenshot:

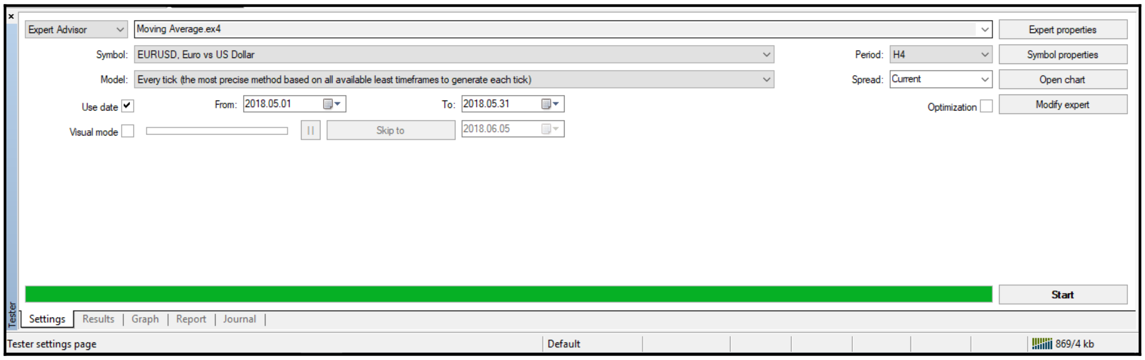


Starting strategy tester

It will open up a vertical tab in the same window and provide you option to select the following:

- **Expert Advisor**: Choose from the list of available advisors, including custom build.
- **Symbol**: List of instruments on which you can apply them.
- **Use date**: Range of dates on from which to which date you want to apply on.
- **Period**: List of available periods, such as M1, M5, M15, or H1.
- **Spread**: List of available spreads to choose from.

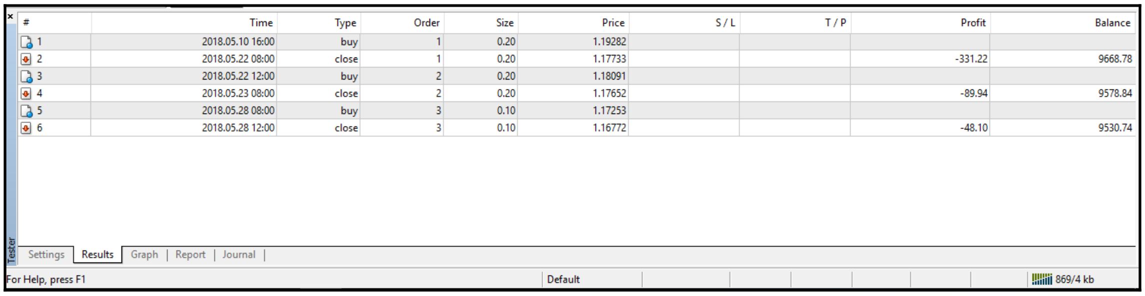
We have selected our required options to start our testing:



Expert Advisor: Moving Average.ex4
Symbol: EURUSD, Euro vs US Dollar
Model: Every tick (the most precise method based on all available least timeframes to generate each tick)
Period: H4
Spread: Current
Use date ✓ From: 2018.05.01 To: 2018.05.31
Visual mode Skip to 2018.06.05
Optimization
Tester settings page

Strategy Tester settings

Now click on the **Start** button to let the strategy test apply your algorithm on historical data and come up with the result. This process will take anywhere from few seconds to couple of minutes, depending on the date you have selected and the algorithm you've applied. Once it finishes, it will display the result in the tabs. The Result tab will display the list of trades that our bot actually performed, as shown in the following screenshot:

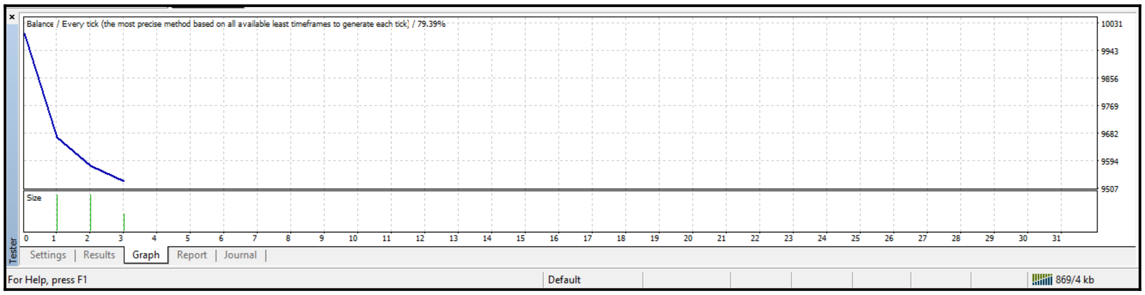


#	Time	Type	Order	Size	Price	S / L	T / P	Profit	Balance
1	2018.05.10 16:00	buy	1	0.20	1.19282				
2	2018.05.22 08:00	close	1	0.20	1.17733			-331.22	9668.78
3	2018.05.22 12:00	buy	2	0.20	1.18091				
4	2018.05.23 08:00	close	2	0.20	1.17652			-89.94	9578.84
5	2018.05.28 08:00	buy	3	0.10	1.17253				
6	2018.05.28 12:00	close	3	0.10	1.16772			-48.10	9530.74

For Help, press F1

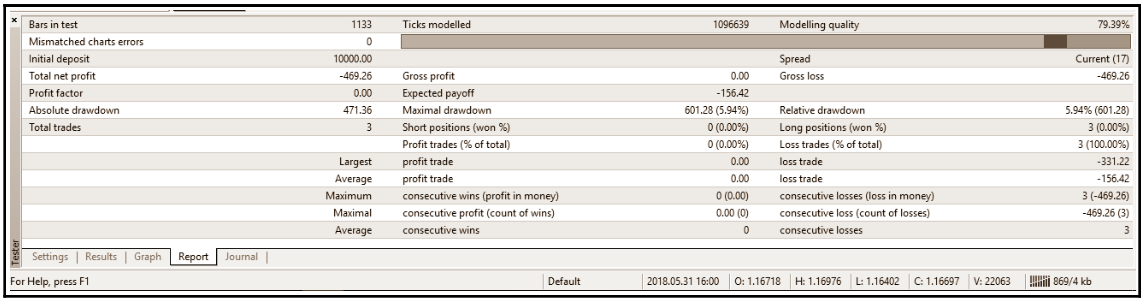
Trades performed by the Bot using Strategy Tester

The **Graph** tab displays the overall indicates of your profit and losses:


Balance / Every tick (the most precise method based on all available least timeframes to generate each tick) / 79.39%
10031
9943
9856
9769
9682
9594
9507
Size
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Tester
Settings | Results | Graph | Report | Journal
For Help, press F1
Default
869/4 kb

Graph generated as a result of Strategy testing

The **Report** tab displays the overall summary of Gross profit, Gross loss, consecutive wins and losses, profit and loss in trade, and so on, similar to the following screenshot:



Bars in test	1133	Ticks modelled	1096639	Modelling quality	79.39%
Mismatched charts errors	0				
Initial deposit	10000.00			Spread	Current (17)
Total net profit	-469.26	Gross profit	0.00	Gross loss	-469.26
Profit factor	0.00	Expected payoff	-156.42		
Absolute drawdown	471.36	Maximal drawdown	601.28 (5.94%)	Relative drawdown	5.94% (601.28)
Total trades	3	Short positions (won %)	0 (0.00%)	Long positions (won %)	3 (0.00%)
		Profit trades (% of total)	0 (0.00%)	Loss trades (% of total)	3 (100.00%)
	Largest	profit trade	0.00	loss trade	-331.22
	Average	profit trade	0.00	loss trade	-156.42
	Maximum	consecutive wins (profit in money)	0 (0.00)	consecutive losses (loss in money)	3 (-469.26)
	Maximal	consecutive profit (count of wins)	0.00 (0)	consecutive loss (count of losses)	-469.26 (3)
	Average	consecutive wins	0	consecutive losses	3

The Report tab displays the summary of the overall progress of Strategy Tester

At this stage, we have successfully performed algorithmic trading using our own custom-built bot and tested it on historical data using the backtesting **Strategy Tester** tool provided in MetaTrader.

Summary

In this chapter, we learned about algorithmic trading and its key benefits. We also learned how big data is influencing the financial market and what are its offering in making it more robust. Then we learned about some of the strategies, such as trend following or the moving-average strategy, arbitrage opportunities, the mean-reversion strategy, the volume-weighted average price, the time-weighted average price, and the percentage of volume strategy, that we can apply to create our own algorithm and strategy to automate trade.

We then jumped into the practical stuff to create our own trade bot, which will be using the mean average strategy to perform trade in the actual market with any third-person involvement. We started by first signing up for a demo trader account. We then downloaded and set up the trading station environment to start writing the code. Then we went through the key definition and the basic structure of the program. After that, we wrote the code to calculate the mean average and apply the strategy so that the program will perform the actual buy and sell when the desired conditions are met.

Then we went through how to backtest our code/Expert Advisor on historical data. It provides many options to choose from, based on your desired input and output. For example, it provides you with the option to select the date range, the period, and other parameters.

In the next chapter, we will discuss the retail recommendation system and how to create one of our own.

19
Retail Recommendation System

More and more consumers are moving away from physical stores and opting for online shopping. The online stores we choose depend on a number of different factors, such as products available, price, discounts, and, the most important of all: user experience. User experience is shaped by the online store's user interface, its support system, its available guidance, and its searching and recommendation systems.

In this chapter, we will explore the following topics:

- Types of recommendation systems:
    - Collaborative
    - Content-based
    - Demographic-based
    - Utility-based
    - Knowledge-based
    - Hybrid model
- Commercially available recommendation systems
- Practical examples of book recommendation systems

Let's start our discussion by first understanding what a recommendation system is, and what different types of recommendation systems exist.

Types of recommendation system

For an online store owner, it is important that they offer their customers different group deals, offers and discounts, and plenty of stock so that customers will stay on-site and make a purchase.

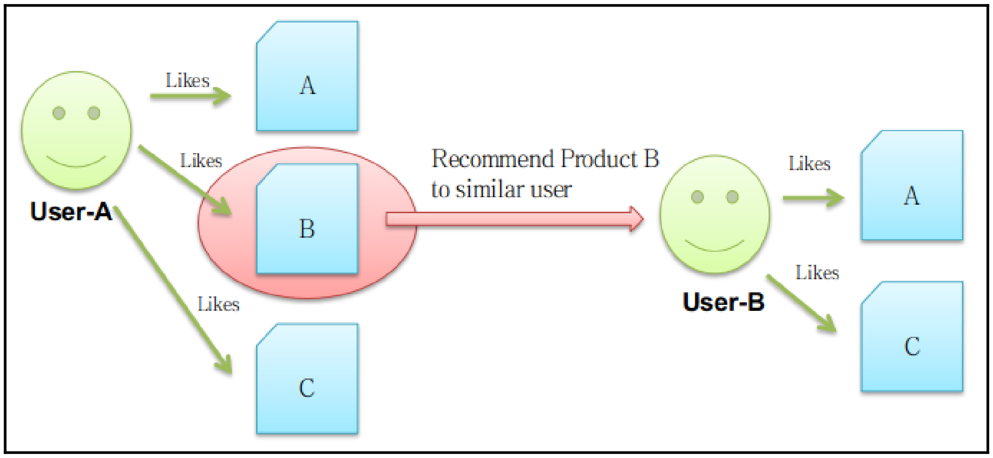
This is not necessarily a way to generate revenue, however. Let's take the example of a good salesperson. If you visit a salesperson and ask them to show you any item, they will not only show you that item but others that you are also likely to buy. Good salespeople create customer profiles using different attributes, such as what the customer is wearing, how they behave, what gadget they are using, and so on. By using a customer profile, customers are more likely to make multiple purchases.

While shopping online, recommendation systems do the job of a good salesperson. A recommendation system will display other items along with the item your user is looking for. For example, YouTube will display a list of other related videos that its users are likely to watch. YouTube often recommends certain videos because of their ratings or content.

Let's now look ahead and see how recommendation works. There are different types of algorithms and mechanisms used in a recommendation system, which we will discuss in the following section.

Collaborative filtering

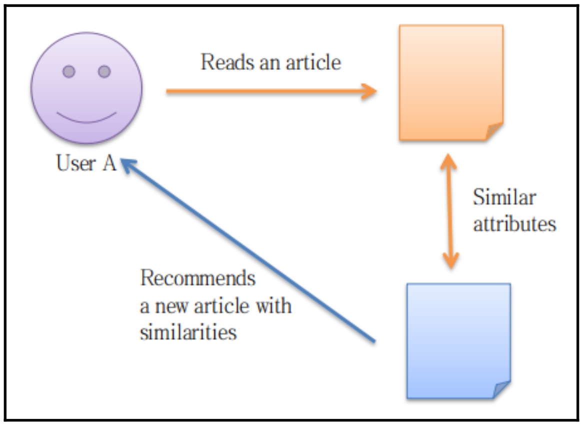
When using **collaborative filtering**, a recommendation system recommends an item-based on the similar interests of others, relying on the ratings and feedback of users. For example, collaborative filtering will flag if user A has purchased product X and user B has purchased products X and Y; the system will then recommend product Y to user A, because another user with similar interests has purchased both products X and Y. The following diagram illustrates the collaborative recommendation mechanism:



Collaborative recommendation mechanism

Content-based filtering

In this content-based filtering, a list of recommended items-based on the attributes of the product that a user is searching for is generated. For example, if one user is currently looking for detergent A, then the recommendation system will display a list of other available detergents, B and C. The following diagram illustrates the content-based filter mechanism:



Content-based filter recommendation mechanism

Demographic-based system

As the name states, this type of recommendation system classifies different items-based on demographic attributes, as generally speaking, different populations like different groups of items. For example, think about your local popular restaurant. Demographic-based filtering mainly relies on recommending a product-based on market research, rather than user rating.

Utility-based system

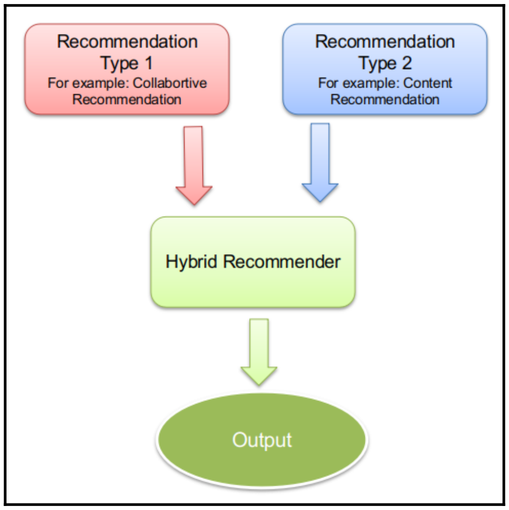
In this type of recommendation system, products are filtered-based on the usefulness of the item for a specific user. Instead of using similar attributes, this filtering depends on unrelated attributes such as vendor reliability, vendor review, product availability, and so on. For example, if user A uses an iPhone, they are more likely to purchase other Apple products.

Knowledge-based system

This type of recommendation system uses user preferences as a benchmark for displaying recommended products. For example, if a user set their preferences so that they want to receive promotions from the electronic department, the recommendation system will display similar items from that category.

Hybrid model

This type of filtering combines the algorithm of two or more recommendation filters that we have already discussed. For example, to recommend relevant products to a user, an online store can apply collaborative filtering to see what other users are also buying, as well as content-based filtering to display alternatives of a product being searched for. The following diagram illustrates the mechanism of the hybrid recommendation system:


Recommendation Type 1
For example: Collabortive Recommendation
Recommendation Type 2
For example: Content Recommendation
Hybrid Recommender
Output

Hybrid recommendation system mechanism

All recommendation systems require continuous review and fine-tuning-based on individual stores so they can more accurately recommend products and subsequently help to boost sales.

Now that we know the different types of recommendation system, let's have a brief look at some of the recommendation systems commercially available at the moment.

Commercial tools

There are many commercially available tools on the market that do the hard job of maintaining and tuning data for you. To use most of these tools, you will have to request demos and get a pricing quote.

Barilliance

Barilliance provides many different types of recommendation services, along with a website recommendation system, personalized emails, and much more. They also offer custom widgets that you can directly apply to your online store. For more information, visit Barilliance's official website at: `http://www.barilliance.com/`.

Softcube

Softcube provides services for user recommendations on websites, along with personalized emails and push notifications. You can opt-in for a 30-day free trial of their service. For more information, visit Softcube's official website at: `http://www.softcube.com/`.

Strands

Strands also provides different algorithms, including collaborative recommendation and content recommendation, along with many different business logic examples and models that you can apply to your algorithm. For more information, visit Strands' official website at `http://retail.strands.com/`.

Monetate

Monetate provides support for your website, emails, push notifications, and mobile applications with different recommendation algorithms that work behind-the-scenes. Monetate also provides support for the integration of algorithms with an existing CRM and POS system for better user experience. For more information, visit Monetate's official website at: `http://www.monetate.com/`.

Nosto

Nosto analyzes user behavior on your online store in real time and recommends product and services accordingly. It also provides tools for applying formulas on real-time user behavior as well as offering on-the-spot recommendations and discounts. Nosto offers a 14-day free trial but requests a percentage share of any revenue it has helped to generate thereafter. For more information, visit Nosto's official website at: `http://www.nosto.com/`.

Let's now move on to something a little more complicated and build our own recommendation system in Python.

Book recommendation system

In this practical section, we will build a recommendation system that will suggest different books to its online users, similar to the system we see on Amazon's website. Before we begin creating a recommendation system, we first need a dataset. Once we have one, we will then analyze that dataset as per our requirements and then apply our findings to recommend different items to our users.

Let's get started by setting up the dataset for our book recommendation system.

Dataset

At this stage, we should fully understand the power of data. If you don't have data, or have very sparse data, than it is almost impossible to build a recommendation system. Note that your data needn't be collected online – it can be the result of market research, user polling, and so on.

In this example, we will use the book rating dataset that was compiled by Cai-Nicolas Ziegler. This dataset contains the data of 270,000 books as rated by 90,000 users. The overall number of ratings it contains is around 1.1 million. The rating scale in this dataset is from 1 to 10, where 1 is the worst and 10 is the best.

The dataset contains the following three CSV files:

- `BX-Books.csv`: This contains information related to the book, such as its title, ISBN number, author, and so on
- `BX-Users.csv`: This contains information related to the user who rated the book, such as user ID, their location, and their age
- `BX-Ratings.csv`: This contains the actual rating given to a book by any user and contains information such as user ID, ISBN number, and rating

This dataset can be download from `http://www2.informatik.uni-freiburg.de/~cziegler/BX/`. It is available in different formats, including SQL dump, CSV dump, and so on. In our example, as discussed, we will be using CSV dump.

Directory structure

Download the dataset and place it in a folder; make sure to extract it in the same folder too. This will now be your main folder, which will eventually contain the code files. In this case, we have placed the folder in `~/Project/RecommendationSystem`.

Now, let's create a new file and save it as `recommender.py` in the same dataset folder. This file will contain the code for our recommendation system.

Code

Open up `recommender.py` in your favorite text editor and start writing the code. We will first import the Python libraries `pandas`, `numpy`, and `matplotlib`. These libraries will help us to move along with our code. Write the following code in your code file:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Reading the dataset

Our next step is to read the information within our CSV dataset and assign it to a variable. Let's start by first exploring information about the books from CSV. The `Books` CSV contains the following data categories in a column format, separated by `;`:

- `ISBN`
- `Book-Title`
- `Book-Author`
- `Year-Of-Publication`
- `Publisher`
- `Image-URL-S`
- `Image-URL-M`
- `Image-URL-L`

Now, we should input the data into our Python program with the following code:

```
books = pd.read_csv('BX-Books.csv', sep=';', error_bad_lines=False,
encoding="latin-1")

books = pd.read_csv('BX-Books.csv', sep=';', error_bad_lines=False,
encoding="latin-1")
books.columns = ['ISBN','Book-Title','Book-Author','Year-Of-
Publication','Publisher','Image-URL-S','Image-URL-M','Image-URL-L']
```

Now that we have read the information related to the books, let's now take a look at the users. The `Users` CSV contains the following columns:

- `User-ID`
- `Location`
- `Age`

We will now read this information by writing the following code in Python, as we did previously:

```
users = pd.read_csv('BX-Users.csv', sep=';', error_bad_lines=False,
encoding="latin-1")

users.columns = ['User-ID','Location','Age']
```

The third and final items we have to look are related to ratings. The `Rating` CSV file contains data in the following columns:

- `User-ID`
- `ISBN`
- `Book-Rating`

We will now read this information by writing the following code in Python, as we did previously:

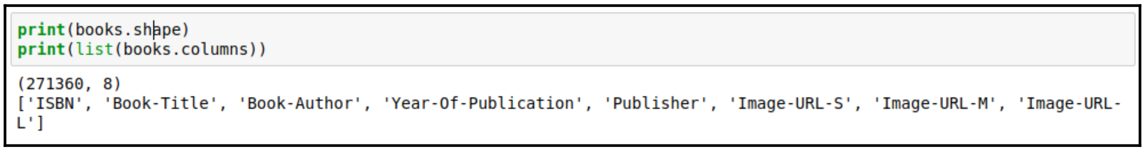
```
ratings = pd.read_csv('BX-Book-Ratings.csv', sep=';',
error_bad_lines=False, encoding="latin-1")

ratings.columns = ['User-ID','ISBN','Book-Rating']
```

Verifying the dataset

We will now verify that all of our data has been loaded successfully. To check this, write the following code to verify we have the data relating to books:

```
print(books.shape)
print(list(books.columns))
```

Once executed, you should see output similar to the following screenshot:



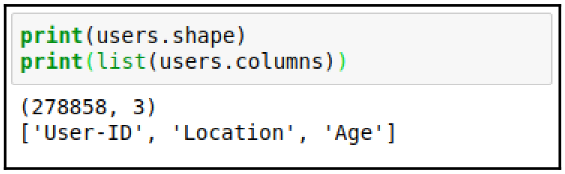
```
print(books.shape)
print(list(books.columns))

(271360, 8)
['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher', 'Image-URL-S', 'Image-URL-M', 'Image-URL-
L']
```

Output of verifying book data

The preceding output tells us there are `271360` pieces of book data available in `8` columns, as well as the column names. Let's now write the following code to verify our data relating to users:

```
print(users.shape)
print(list(users.columns))
```

Once executed, you should see output similar to the following screenshot:



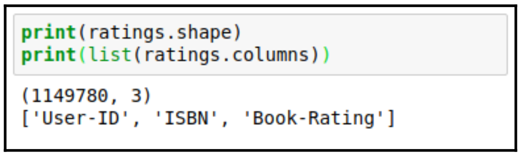
```
print(users.shape)
print(list(users.columns))

(278858, 3)
['User-ID', 'Location', 'Age']
```

Output of verifying user data

The preceding output tells us there are `278858` pieces of user data available in `3` columns. Last but not least, let's now verify the data relating to ratings with the following code:

```
print(ratings.shape)
print(list(ratings.columns))
```

Once executed, you should see output similar to the following screenshot:



```
print(ratings.shape)
print(list(ratings.columns))

(1149780, 3)
['User-ID', 'ISBN', 'Book-Rating']
```

Output of verifying rating data

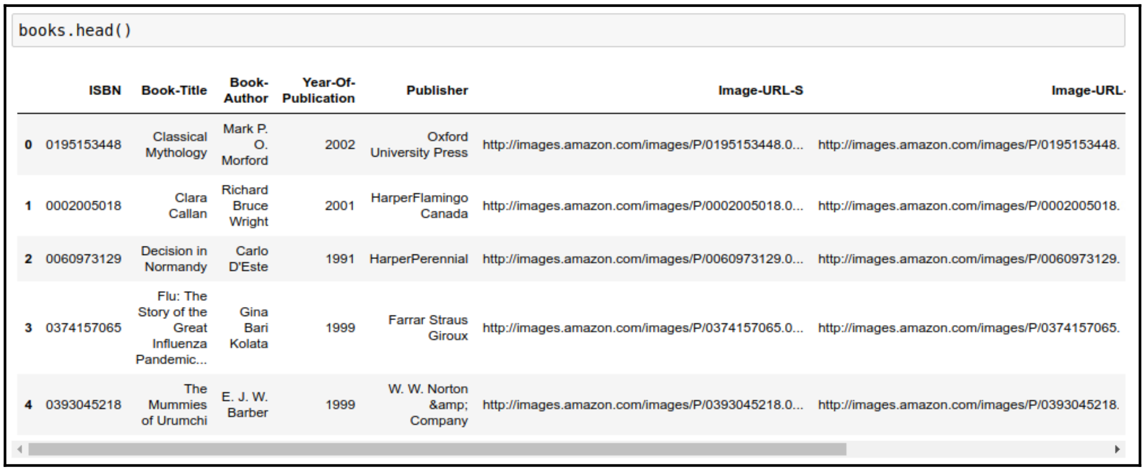
The preceding output tells us there are `1149780` pieces of rating data available for different books in our dataset; the data has been divided into `3` columns.

In this example, we have used *Jupyter Notebook*. Jupyter Notebook is a great tool for getting line-by-line output and efficiently uses data in memory. There are many benefits of using Jupyter, particularly if you plan on a hit and trail approach in your program. For more information, visit Jupyter's official website at: `http://jupyter.org/`.

If you are using Jupyter Notebook, write the following code to see the books data loaded from the dataset:

```
books.head()
```

Once executed, you should see output similar to the following screenshot:



```
books.head()
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0374157065.
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton &amp; Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0393045218.

You can write the same command for both user and rating data if you want to view the load information for the CSV file in our program.

Data analysis

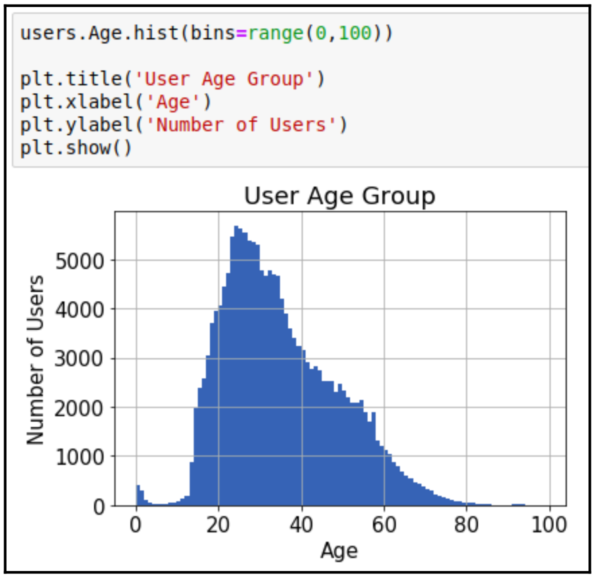
For data analysis, we will be using the `matplotlib` library to display the output in an easy-to-read way. We learned how to use *Matplotlib* in `Chapter 17`, *Data Visualization*, so let's move on and start analyzing our data.

Age group

In order to analyze our data in a visual form, we will apply `matplotlib` to our user dataset by writing the following code:

```
users.Age.hist(bins=range(0,100))

plt.title('User Age Group')
plt.xlabel('Age')
plt.ylabel('Number of Users')
plt.show()
```

The following screenshot displays the output of the preceding program:



```
users.Age.hist(bins=range(0,100))

plt.title('User Age Group')
plt.xlabel('Age')
plt.ylabel('Number of Users')
plt.show()
```

Visual display of users' age groups

As you can see in the preceding output, users who participated in collecting this data were mainly in the 20 to 40 age group.

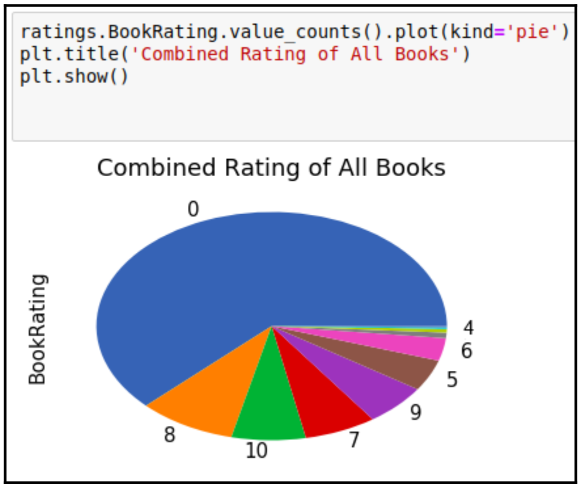
Commutative rating

Let's now plot a graph to display the commutative rating list of all books. To do this, write the following code in your program:

```
ratings.BookRating.value_counts().plot(kind='pie')
```

```
plt.title('Combined Rating of All Books')
plt.xlabel('Rating')
plt.show()
```

Once executed, you should see something similar to the following screenshot:



```
ratings.BookRating.value_counts().plot(kind='pie')
plt.title('Combined Rating of All Books')
plt.show()
```

The combined rating of all books

If we review the result, as we can see in the preceding diagram, we can see that less than 5% of the books available are the most popular books.

Note that you can choose how you visualize your data, and how you extract meaningful information from it. Graphs and pie charts are just two examples of this. Let's now move on and apply different filters and algorithms so that we can start recommending books to users-based on different criteria.

Algorithms

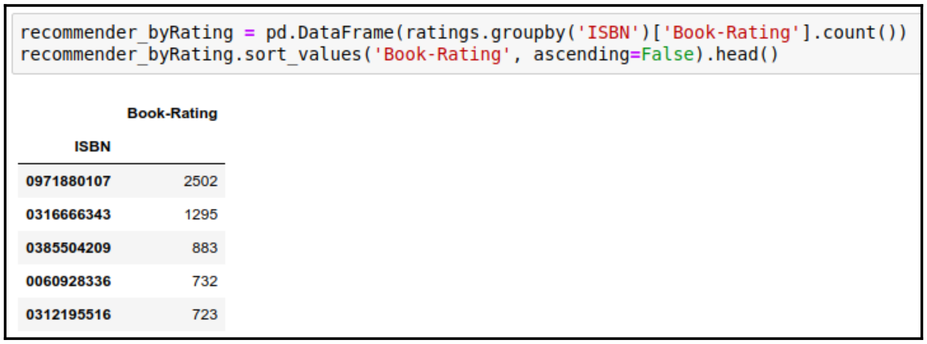
We will now take a look at how to recommend books to users-based on different recommendation scenarios. First of all, let's take a very simple scenario and start recommending books that have received the best ratings.

Top-rated books

Now that we have a large dataset, we want to be able to recommend all of the top-rated books to our users. To do this, we will group our data based on the ISBN number, which is unique to reach book. Write the following code in your program to do the same:

```
recommender_byRating = pd.DataFrame(ratings.groupby('ISBN')['Book-
Rating'].count())

recommender_byRating.sort_values('Book-Rating', ascending=False).head()
```

Once executed, you should see output similar to the following screenshot:



```
recommender_byRating = pd.DataFrame(ratings.groupby('ISBN')['Book-Rating'].count())
recommender_byRating.sort_values('Book-Rating', ascending=False).head()
```

	Book-Rating
ISBN	
0971880107	2502
0316666343	1295
0385504209	883
0060928336	732
0312195516	723

The top-rated books in our dataset

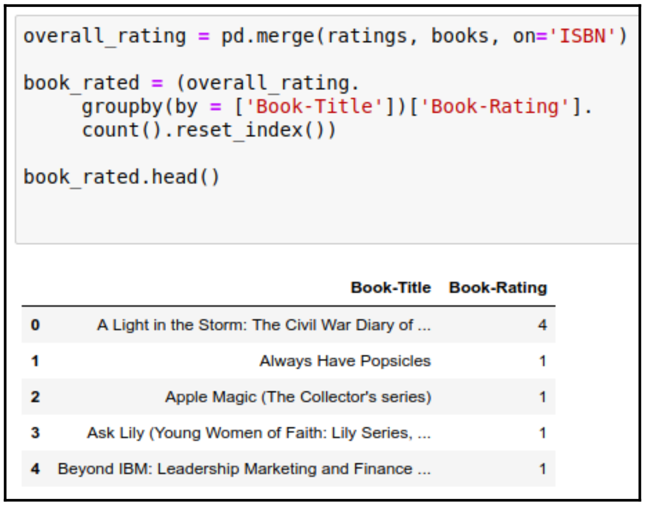
The preceding output displays the top five books that received the most ratings by users. It is worth noting that here we are not seeing books with the highest ratings, but rather the books that have received the most ratings.

Popular books

Let's now find out the most popular book so we can recommend it to our users. To do this, we will merge the datasets of both books and ratings and treat them as one. This will help us to apply different conditions such as group by and count with more ease. Write the following code to figure out the most popular books in our dataset:

```
overall_rating = pd.merge(ratings, books, on='ISBN')

book_rated = (overall_rating.groupby(by = ['Book-Title'])['Book-
Rating'].count().reset_index())

book_rated.head()
```

Once executed, you should see output similar to the following screenshot:



```
overall_rating = pd.merge(ratings, books, on='ISBN')

book_rated = (overall_rating.
     groupby(by = ['Book-Title'])['Book-Rating'].
     count().reset_index())

book_rated.head()
```

	Book-Title	Book-Rating
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Ask Lily (Young Women of Faith: Lily Series, ...	1
4	Beyond IBM: Leadership Marketing and Finance ...	1

Popular books-based on rating

Demographic-based recommendation

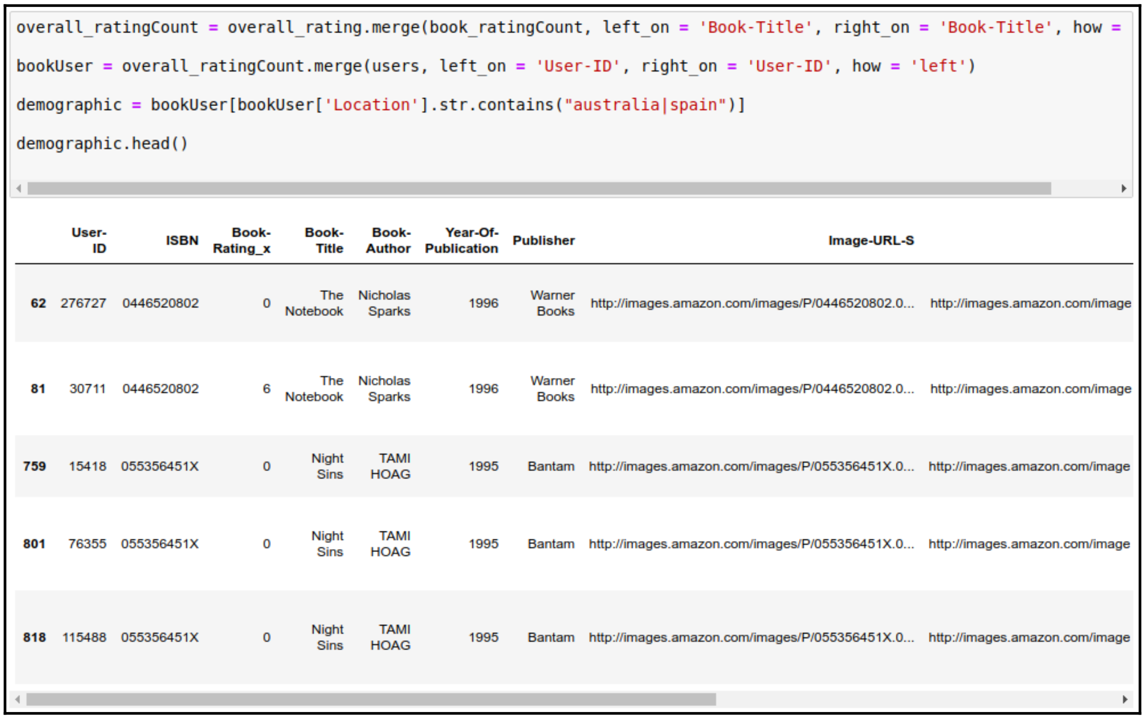
We will now recommend the top-rated books to a user from `australia` or `spain`. To do this, we will combine the books and ratings datasets along with the `User` dataset. Write the following code in order to do the same:

```
overall_ratingCount = overall_rating.merge(book_ratingCount, left_on =
'Book-Title', right_on = 'Book-Title', how = 'left')

bookUser = overall_ratingCount.merge(users, left_on = 'User-ID', right_on =
'User-ID', how = 'left')
```

We now need to filter our result for users residing in `australia` and `spain` and print them; we do this by writing the following lines of code:

```
demographic =
bookUser[bookUser['Location'].str.contains("australia|spain")]

demographic.head()
```

Once executed, you should see output similar to the following screenshot:



```
overall_ratingCount = overall_rating.merge(book_ratingCount, left_on = 'Book-Title', right_on = 'Book-Title', how =

bookUser = overall_ratingCount.merge(users, left_on = 'User-ID', right_on = 'User-ID', how = 'left')

demographic = bookUser[bookUser['Location'].str.contains("australia|spain")]

demographic.head()
```

	User-ID	ISBN	Book-Rating_x	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
62	276727	0446520802	0	The Notebook	Nicholas Sparks	1996	Warner Books	http://images.amazon.com/images/P/0446520802.0...	http://images.amazon.com/image
81	30711	0446520802	6	The Notebook	Nicholas Sparks	1996	Warner Books	http://images.amazon.com/images/P/0446520802.0...	http://images.amazon.com/image
759	15418	055356451X	0	Night Sins	TAMI HOAG	1995	Bantam	http://images.amazon.com/images/P/055356451X.0...	http://images.amazon.com/image
801	76355	055356451X	0	Night Sins	TAMI HOAG	1995	Bantam	http://images.amazon.com/images/P/055356451X.0...	http://images.amazon.com/image
818	115488	055356451X	0	Night Sins	TAMI HOAG	1995	Bantam	http://images.amazon.com/images/P/055356451X.0...	http://images.amazon.com/image

Demographic recommendation code and output

The preceding output shows us what users residing in `australia` and `spain` have rated as their favorite books, so that we can recommend users-based on their `Location` relevant books. Bear in mind that we have covered basic algorithms to get you started. You may want to fine-tune what we've already looked at to produce more advanced output that is based on precise requirements.

Useful resources

Most companies don't like to discard data even if they are not using it – after all, data is powerful. You can therefore find generalized data available to the public for research purposes. You can find many of these datasets freely available to download for use in your projects on the internet.

Yahoo also provides many different datasets, and they have a dedicated website where you can find them at: `https://webscope.sandbox.yahoo.com/`. The main categories for which Yahoo provides datasets are as follows:

- **Advertising and market data**
- **Competition data**
- **Computing systems data**
- **Graph and social data**
- **Image data**
- **Language data**
- **Ratings and classification data**

As an example, if you browse through **rating and classification data**, you will find the following datasets:

- **Music User Ratings of Musical Artists**
- **Music User Ratings of Songs with Artist, Album, and Genre Meta Information**
- **Music ratings for User Selected and Randomly Selected songs**
- **Movies User Ratings and Descriptive Content Information**
- **Delicious Popular URLs and Tags**

For researchers like us, it's vital that we have access to data that we can apply different algorithms to and analyze different patterns in. Recognizing the importance of this, Netflix organized a competition asking users to improve their movie recommendation system. The winning algorithm ended up improving Netflix's recommendation system by 10%.

Summary

In this chapter, we learned about what a recommendation system is, including the different types of recommendation systems, such as collaborative-based, content-based, demographic-based, utility-based, knowledge-based, and a hybrid approach. We then looked at different tools that are commercially available that have more advanced algorithms that can boost your store's user experience and sales.

We then moved on to something more practical: building our own book recommendation system. We built our model-based on three datasets: books, users, and ratings. We started off by loading the dataset in our program and verifying it by outputting different print statements related to a dataset. We also used the `matplotlib` library to visually analyze the data. We then applied different algorithms in order to recommend books to our users, such as top-rated books, the most popular books, the top-rated books according to location, and so on.

Finally, we looked at some useful datasets available online that are similar to the ones used in this chapter.

Other Books You May Enjoy

Index