The Data Engineering Cookbook

Mastering The Plumbing Of Data Science

Andreas Kretz

September 12, 2019

v3.0

Contents

I Introduction 10

1 How To Use This Cookbook 11

2 Data Engineer vs Data Scientist 12

2.1 Data Scientist . . . 12

2.2 Data Engineer . . . 13

2.3 Who Companies Need . . . 14

II Basic Data Engineering Skills 16

3 Learn To Code 17

4 Get Familiar With Git 18

5 Agile Development 19

5.1 Why is agile so important? . . . 19

5.2 Agile rules I learned over the years . . . 20

5.2.1 Is the method making a difference? . . . 20

5.2.2 The problem with outsourcing . . . 20

5.2.3 Knowledge is king: A lesson from Elon Musk . . . 21

5.2.4 How you really can be agile . . . 21

5.3 Agile Frameworks . . . 22

5.3.1 Scrum . . . 22

5.3.2 OKR . . . 22

5.4 Software Engineering Culture . . . 22

6 Learn how a Computer Works 24

6.1 CPU,RAM,GPU,HDD . . . 24

6.2 Differences between PCs and Servers . . . 24

7 Computer Networking - Data Transmission **25**
7.1 OSI Model . . . 25
7.2 IP Subnetting . . . 25
7.3 Switch, Level 3 Switch . . . 26
7.4 Router . . . 26
7.5 Firewalls . . . 26

8 Security and Privacy **27**
8.1 SSL Public & Private Key Certificates . . . 27
8.2 What is a certificate authority . . . 27
8.3 JSON Web Tokens . . . 27
8.4 GDPR regulations . . . 27
8.5 Privacy by design . . . 27

9 Linux **29**
9.1 OS Basics . . . 29
9.2 Shell scripting . . . 29
9.3 Cron jobs . . . 29
9.4 Packet management . . . 30

10 The Cloud **31**
10.1 IaaS vs PaaS vs SaaS . . . 31
10.2 AWS, Azure, IBM, Google . . . 31
10.2.1 AWS . . . 31
10.2.2 Azure . . . 32
10.2.3 IBM . . . 32
10.2.4 Google . . . 32
10.3 Cloud vs On-Premises . . . 32
10.4 Security . . . 32
10.5 Hybrid Clouds . . . 32

11 Security Zone Design **33**
11.1 How to secure a multi layered application . . . 33
11.2 Cluster security with Kerberos . . . 33

12 Big Data **34**
12.1 What is big data and where is the difference to data science and data analytics? . . . 34
12.2 The 4 Vs of Big Data . . . 34

12.3 Why Big Data? . . . 35
12.3.1 Planning is Everything . . . 36
12.3.2 The problem with ETL . . . 36
12.3.3 Scaling Up . . . 37
12.3.4 Scaling Out . . . 38
12.3.5 Please don't go Big Data . . . 39

13 My Big Data Platform Blueprint **40**
13.1 Ingest . . . 41
13.2 Analyse / Process . . . 41
13.3 Store . . . 42
13.4 Display . . . 43

14 Lambda Architecture **44**
14.1 Batch Processing . . . 44
14.2 Stream Processing . . . 45
14.3 Should you do stream or batch processing? . . . 46
14.4 Lambda Architecture Alternative . . . 46
14.4.1 Kappa Architecture . . . 46
14.4.2 Kappa Architecture with Kudu . . . 46
14.5 Why a Good Data Platform Is Important . . . 46

15 Data Warehouse vs Data Lake **47**

16 Hadoop Platforms **48**
16.1 What is Hadoop . . . 48
16.2 What makes Hadoop so popular? . . . 49
16.3 Hadoop Ecosystem Components . . . 50
16.4 Hadoop Is Everywhere? . . . 51
16.5 Should you learn Hadoop? . . . 52
16.6 How does a Hadoop System architecture look like . . . 52
16.7 What tools are usually in a with Hadoop Cluster . . . 52
16.8 How to select Hadoop Cluster Hardware . . . 52

17 Docker **53**
17.1 What is docker and what do you use it for . . . 53
17.1.1 Don't Mess Up Your System . . . 53
17.1.2 Preconfigured Images . . . 53
17.1.3 Take It With You . . . 54
17.2 Kubernetes Container Deployment . . . 54

17.3 How to create, start, stop a Container . . . 55
17.4 Docker micro services? . . . 55
17.5 Kubernetes . . . 55
17.6 Why and how to do Docker container orchestration . . . 55
17.7 Useful Docker Commands . . . 55

18 REST APIs **57**
18.1 API Design . . . 57
18.2 Implementation Frameworks . . . 57
18.3 OAuth security . . . 58

19 Databases **59**
19.1 SQL Databases . . . 59
19.1.1 PostgreSQL DB . . . 59
19.1.2 Database Design . . . 59
19.1.3 SQL Queries . . . 59
19.1.4 Stored Procedures . . . 59
19.1.5 ODBC/JDBC Server Connections . . . 59
19.2 NoSQL Stores . . . 59
19.2.1 KeyValue Stores (HBase) . . . 59
19.2.2 Document Store HDFS . . . 59
19.2.3 Document Store MongoDB . . . 62
19.2.4 Elasticsearch Search Engine and Document Store . . . 63
19.2.5 Hive Warehouse . . . 64
19.2.6 Impala . . . 64
19.2.7 Kudu . . . 64
19.2.8 Apache Druid . . . 64
19.2.9 InfluxDB Time Series Database . . . 64
19.2.10 MPP Databases (Greenplum) . . . 65

20 Data Processing and Analytics - Frameworks **66**
20.1 Is ETL still relevant for Analytics? . . . 66
20.2 Stream Processing . . . 66
20.2.1 Three methods of streaming . . . 66
20.2.2 At Least Once . . . 67
20.2.3 At Most Once . . . 67
20.2.4 Exactly Once . . . 67
20.2.5 Check The Tools! . . . 68
20.3 MapReduce . . . 68
20.3.1 How does MapReduce work . . . 69

20.3.2 Example . . . 70
20.3.3 What is the limitation of MapReduce? . . . 72
20.4 Apache Spark . . . 72
20.4.1 What is the difference to MapReduce? . . . 72
20.4.2 How does Spark fit to Hadoop? . . . 73
20.4.3 Where's the difference? . . . 73
20.4.4 Spark and Hadoop is a perfect fit . . . 74
20.4.5 Spark on YARN: . . . 74
20.4.6 My simple rule of thumb: . . . 75
20.4.7 Available Languages . . . 75
20.4.8 How Spark works: Driver, Executor, Sparkcontext . . . 75
20.4.9 Spark batch vs stream processing . . . 76
20.4.10 How does Spark use data from Hadoop . . . 76
20.4.11 What are RDDs and how to use them . . . 76
20.4.12 How and why to use SparkSQL? . . . 77
20.4.13 What are DataFrames how to use them . . . 77
20.4.14 Machine Learning on Spark? (Tensor Flow) . . . 77
20.4.15 MLlib: . . . 78
20.4.16 Spark Setup . . . 78
20.4.17 Spark Resource Management . . . 78
20.5 Apache Nifi . . . 79
20.6 StreamSets . . . 80

21 Apache Kafka **81**
21.1 Why a message queue tool? . . . 81
21.2 Kafka architecture . . . 81
21.3 What are topics . . . 81
21.4 What does Zookeeper have to do with Kafka . . . 81
21.5 How to produce and consume messages . . . 81
21.6 KAFKA Commands . . . 81

22 Machine Learning **83**
22.1 How to do Machine Learning in production . . . 83
22.2 Why machine learning in production is harder then you think . . . 83
22.3 Models Do Not Work Forever . . . 84
22.4 Where The Platforms That Support This? . . . 84
22.5 Training Parameter Management . . . 84
22.6 What's Your Solution? . . . 85
22.7 How to convince people machine learning works . . . 85

22.8 No Rules, No Physical Models . . . 85
22.9 You Have The Data. USE IT! . . . 86
22.10 Data is Stronger Than Opinions . . . 86
22.11 AWS Sagemaker . . . 87

23 Data Visualization **88**
23.1 Android & IOS . . . 88
23.2 How to design APIs for mobile apps . . . 88
23.3 How to use Webservers to display content . . . 88
23.3.1 Tomcat . . . 89
23.3.2 Jetty . . . 89
23.3.3 NodeRED . . . 89
23.3.4 React . . . 89
23.4 Business Intelligence Tools . . . 89
23.4.1 Tableau . . . 89
23.4.2 PowerBI . . . 89
23.4.3 Quliksense . . . 89
23.5 Identity & Device Management . . . 89
23.5.1 What is a digital twin? . . . 89
23.5.2 Active Directory . . . 89

III Data Engineering Course: Building A Data Platform **90**

24 What We Want To Do **91**

25 Thoughts On Choosing A Development Environment **92**

26 A Look Into the Twitter API **93**

27 Ingesting Tweets with Apache Nifi **94**

28 Writing from Nifi to Apache Kafka **95**

29 Apache Zeppelin **96**
29.1 Install and Ingest Kafka Topic . . . 96
29.2 Processing Messages with Spark & SparkSQL . . . 96
29.3 Visualizing Data . . . 96

30 Switch Processing from Zeppelin to Spark **97**
30.1 Install Spark . . . 97
30.2 Ingest Messages from Kafka . . . 97

30.3 Writing from Spark to Kafka . . . 97
30.4 Move Zeppelin Code to Spark . . . 97

IV Case Studies 98

31 How I do Case Studies 99
31.1 Data Science @Airbnb . . . 99
31.2 Data Science @Amazon . . . 99
31.3 Data Science @Baidu . . . 99
31.4 Data Science @Blackrock . . . 100
31.5 Data Science @BMW . . . 100
31.6 Data Science @Booking.com . . . 100
31.7 Data Science @CERN . . . 100
31.8 Data Science @Disney . . . 101
31.9 Data Science @DLR . . . 101
31.10Data Science @Drivetribe . . . 101
31.11Data Science @Dropbox . . . 102
31.12Data Science @Ebay . . . 102
31.13Data Science @Expedia . . . 102
31.14Data Science @Facebook . . . 102
31.15Data Science @Google . . . 102
31.16Data Science @Grammarly . . . 102
31.17Data Science @ING Fraud . . . 102
31.18Data Science @Instagram . . . 103
31.19Data Science @LinkedIn . . . 103
31.20Data Science @Lyft . . . 103
31.21Data Science @NASA . . . 104
31.22Data Science @Netflix . . . 104
31.23Data Science @OLX . . . 108
31.24Data Science @OTTO . . . 108
31.25Data Science @Paypal . . . 108
31.26Data Science @Pinterest . . . 108
31.27Data Science @Salesforce . . . 109
31.28Data Science @Siemens Mindsphere . . . 109
31.29Data Science @Slack . . . 110
31.30Data Science @Spotify . . . 110
31.31Data Science @Symantec . . . 110
31.32Data Science @Tinder . . . 110

31.33Data Science @Twitter 111
31.34Data Science @Uber 111
31.35Data Science @Upwork 112
31.36Data Science @Woot 112
31.37Data Science @Zalando 112

V 1001 Data Engineering Interview Questions 114

32 Live Streams 116

33 All Interview Questions 117

Part I

Introduction

1 How To Use This Cookbook

What do you actually need to learn to become an awesome data engineer? Look no further, you'll find it here.

If you are looking for AI algorithms and such data scientist things, this book is not for you.

How to use this document:
First of all, this is not a training! This cookbook is a collection of skills that I value highly in my daily work as a data engineer. It's intended to be a starting point for you to find the topics to look into and become an awesome data engineer.

You are going to find ***Five Types of Content*** in this book: Articles I wrote, links to my podcast episodes (video & audio), more than 200 links to helpful websites I like, data engineering interview questions and case studies.

This book is a work in progress!
As you can see, this book is not finished. I'm constantly adding new stuff and doing videos for the topics. But obviously, because I do this as a hobby my time is limited. You can help making this book even better.

Help make this book awesome!
If you have some cool links or topics for the cookbook, please become a contributor on GitHub: https://github.com/andkret/Cookbook. Pull the repo, add them and create a pull request. Or join the discussion by opening Issues. You can also write me an email any time to plumbersofdatascience@gmail.com. Tell me your thoughts, what you value, what you think should be included, or correct me where I am wrong.

This Cookbook is and will always be free!
I don't want to sell you this book, but please support what you like and join my Patreon: https://www.patreon.com/plumbersofds

Check out this podcast episode where I talk in detail why I decided to share all this information for free: #079 Trying to stay true to myself and making the cookbook public on GitHub

2 Data Engineer vs Data Scientist

Podcast Episode: #050 Data Engineer, Scientist or Analyst - Which One Is For You?	
In this podcast we talk about the differences between data scientists, analysts and engineers. Which are the three main data science jobs. All three are super important. This makes it easy to decide	
YouTube	Click here to watch
Audio	Click here to listen

Table 2.1: Podcast: 050 Data Engineer, Scientist or Analyst - Which One Is For You?

2.1 Data Scientist

Data scientists aren't like every other scientist.

Data scientists do not wear white coats or work in high tech labs full of science fiction movie equipment. They work in offices just like you and me.

What differs them from most of us is that they are math experts. They use linear algebra and multivariable calculus to create new insight from existing data.

How exactly does this insight look?

Here's an example:

An industrial company produces a lot of products that need to be tested before shipping.

Usually such tests take a lot of time because there are hundreds of things to be tested. All to make sure that your product is not broken.

Wouldn't it be great to know early if a test fails ten steps down the line? If you knew that you could skip the other tests and just trash the product or repair it.

That's exactly where a data scientist can help you, big-time. This field is called predictive analytics and the technique of choice is machine learning.

Machine what? Learning?

Yes, machine learning, it works like this:

You feed an algorithm with measurement data. It generates a model and optimises it based on the data you fed it with. That model basically represents a pattern of how your data is looking. You show that model new data and the model will tell you if the data still represents the data you have trained it with. This technique can also be used for predicting machine failure in advance with machine learning. Of course the whole process is not that simple.

The actual process of training and applying a model is not that hard. A lot of work for the data scientist is to figure out how to pre-process the data that gets fed to the algorithms.

In order to train a algorithm you need useful data. If you use any data for the training the produced model will be very unreliable.

A unreliable model for predicting machine failure would tell you that your machine is damaged even if it is not. Or even worse: It would tell you the machine is ok even when there is an malfunction.

Model outputs are very abstract. You also need to post-process the model outputs to receive health values from 0 to 100.

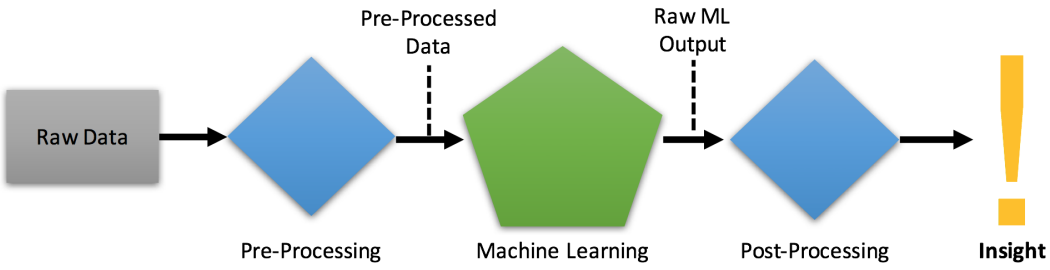

Figure 2.1: The Machine Learning Pipeline

2.2 Data Engineer

Data Engineers are the link between the management's big data strategy and the data scientists that need to work with data.

What they do is building the platforms that enable data scientists to do their magic.

These platforms are usually used in five different ways:

- Data ingestion and storage of large amounts of data

- Algorithm creation by data scientists
- Automation of the data scientist's machine learning models and algorithms for production use
- Data visualisation for employees and customers
- Most of the time these guys start as traditional solution architects for systems that involve SQL databases, web servers, SAP installations and other "standard" systems.

But to create big data platforms the engineer needs to be an expert in specifying, setting up and maintaining big data technologies like: Hadoop, Spark, HBase, Cassandra, MongoDB, Kafka, Redis and more.

What they also need is experience on how to deploy systems on cloud infrastructure like at Amazon or Google or on-premise hardware.

Podcast Episode: #048 From Wannabe Data Scientist To Engineer My Journey	
In this episode Kate Strachnyi interviews me for her humans of data science podcast. We talk about how I found out that I am more into the engineering part of data science.	
YouTube	Click here to watch
Audio	Click here to listen

Table 2.2: Podcast: 048 From Wannabe Data Scientist To Engineer My Journey

2.3 Who Companies Need

For a good company it is absolutely important to get well trained data engineers and data scientists. Think of the data scientist as the professional race car driver. A fit athlete with talent and driving skills like you have never seen.

What he needs to win races is someone who will provide him the perfect race car to drive. That's what the solution architect is for.

Like the driver and his team the data scientist and the data engineer need to work closely together. They need to know the different big data tools inside and out.

That's why companies are looking for people with Spark experience. It is a common ground between both that drives innovation.

Spark gives data scientists the tools to do analytics and helps engineers to bring the data scientist's algorithms into production. After all, those two decide how good the data

platform is, how good the analytics insight is and how fast the whole system gets into a production ready state.

Part II

Basic Data Engineering Skills

3 Learn To Code

Why this is important: Without coding you cannot do much in data engineering. I cannot count the number of times I needed a quick Java hack.

The possibilities are endless:

- Writing or quickly getting some data out of a SQL DB
- Testing to produce messages to a Kafka topic
- Understanding the source code of a Java Webservice
- Reading counter statistics out of a HBase key value store

So, which language do I recommend then?

I highly recommend Java. It's everywhere!

When you are getting into data processing with Spark you should use Scala. But, after learning Java this is easy to do.

Also Python is a great choice. It is super versatile.

Personally however, I am not that big into Python. But I am going to look into it

Where to Learn? There's a Java Course on Udemy you could look at: https://www.udemy.com/java-programming-tutorial-for-beginners

- OOP Object oriented programming
- What are Unit tests to make sure what you code is working
- Functional Programming
- How to use build management tools like Maven
- Resilient testing (?)

I talked about the importance of learning by doing in this podcast: https://anchor.fm/andreaskayy/episodes/Learning-By-Doing-Is-The-Best-Thing-Ever---PoDS-035-e25g44

4 Get Familiar With Git

Why this is important: One of the major problems with coding is to keep track of changes. It is also almost impossible to maintain a program you have multiple versions of.

Another problem is the topic of collaboration and documentation, which is super important.

Let's say you work on a Spark application and your colleagues need to make changes while you are on holiday. Without some code management they are in huge trouble:

Where is the code? What have you changed last? Where is the documentation? How do we mark what we have changed?

But if you put your code on GitHub your colleagues can find your code. They can understand it through your documentation (please also have in-line comments)

Developers can pull your code, make a new branch and do the changes. After your holiday you can inspect what they have done and merge it with your original code and you end up having only one application.

Where to learn: Check out the GitHub Guides page where you can learn all the basics: https://guides.github.com/introduction/flow/

This great GitHub commands cheat sheet saved my butt multiple times: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

Also look into:

- Pull
- Push
- Branching
- Forking

5 Agile Development

Agility, the ability to adapt quickly to changing circumstances.

These days everyone wants to be agile. Big or small company people are looking for the "startup mentality".

Many think it's the corporate culture. Others think it's the process how we create things that matters.

In this article I am going to talk about agility and self-reliance. About how you can incorporate agility in your professional career.

5.1 Why is agile so important?

Historically development is practiced as a hard defined process. You think of something, specify it, have it developed and then built in mass production.

It's a bit of an arrogant process. You assume that you already know exactly what a customer wants. Or how a product has to look and how everything works out.

The problem is that the world does not work this way!

Often times the circumstances change because of internal factors.

Sometimes things just do not work out as planned or stuff is harder than you think.

You need to adapt.

Other times you find out that you build something customers do not like and need to be changed.

You need to adapt.

That's why people jump on the Scrum train. Because Scrum is the definition of agile development, right?

5.2 Agile rules I learned over the years

5.2.1 Is the method making a difference?

Yes, Scrum or Google's OKR can help to be more agile. The secret to being agile however, is not only how you create.

What makes me cringe is people trying to tell you that being agile starts in your head. So, the problem is you?

No!

The biggest lesson I have learned over the past years is this: Agility goes down the drain when you outsource work.

5.2.2 The problem with outsourcing

I know on paper outsourcing seems like a no-brainer: Development costs against the fixed costs.

It is expensive to bind existing resources on a task. It is even more expensive if you need to hire new employees.

The problem with outsourcing is that you pay someone to build stuff for you.

It does not matter who you pay to do something for you. He needs to make money.

His agenda will be to spend as less time as possible on your work. That is why outsourcing requires contracts, detailed specifications, timetables and delivery dates.

He doesn't want to spend additional time on a project, only because you want changes in the middle. Every unplanned change costs him time and therefore money.

If so, you need to make another detailed specification and a contract change.

He is not going to put his mind into improving the product while developing. Firstly because he does not have the big picture. Secondly because he does not want to.

He is doing as he is told.

Who can blame him? If I was the subcontractor I would do exactly the same!

Does this sound agile to you?

5.2.3 Knowledge is king: A lesson from Elon Musk

Doing everything in house, that's why startups are so productive. No time is wasted on waiting for someone else.

If something does not work, or needs to be changed, there is someone in the team who can do it right away.

One very prominent example who follows this strategy is Elon Musk.

Tesla's Gigafactories are designed to get raw materials in on one side and spit out cars on the other. Why do you think Tesla is building Gigafactories who cost a lot of money?

Why is SpaceX building its one space engines? Clearly there are other, older, companies who could do that for them.

Why is Elon building tunnel boring machines at his new boring company?

At first glance this makes no sense!

5.2.4 How you really can be agile

If you look closer it all comes down to control and knowledge. You, your team, your company, needs to do as much as possible on your own. Self-reliance is king.

Build up your knowledge and therefore the teams knowledge. When you have the ability to do everything yourself, you are in full control.

You can build electric cars, rocket engines or bore tunnels.

Don't largely rely on others and be confident to just do stuff on your own.

Dream big and JUST DO IT!

PS. Don't get me wrong. You can still outsource work. Just do it in a smart way by outsourcing small independent parts.

5.3 Agile Frameworks

5.3.1 Scrum

There's a interesting Scrum Medium publication with a lot of details about Scrum: https://medium.com/serious-scrum

Also this scrum guide webpage has good infos about Scrum: https://www.scrumguides.org/scrum-guide.html

5.3.2 OKR

I personally love OKR, been doing it for years. Especially for smaller teams OKR is great. You don't have a lot of overhead and get work done. It helps you stay focused and look at the bigger picture.

I recommend to do a sync meeting every Monday. There you talk about what happened last week and what you are going to work on this week.

I talked about this in this Podcast: https://anchor.fm/andreaskayy/embed/episodes/Agile-Development-Is-Important-But-Please-Dont-Do-Scrum--PoDS-041-e2e2j4

This is also this awesome 1,5 hours startup guide from Google: https://youtu.be/mJB83EZtAjc I really love this video, I rewatched it multiple times.

5.4 Software Engineering Culture

The software engineering and development culture is super important. How does a company handle product development with hundreds of developers. Check out this podcast:

Podcast Episode: #070 Engineering Culture At Spotify	
In this podcast we look at the engineering culture at Spotify, my favorite music streaming service. The process behind the development of Spotify is really awesome.	
YouTube	Click here to watch
Audio	Click here to listen

Table 5.1: Podcast: 070 Engineering Culture At Spotify

Some interesting slides:

https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

https://labs.spotify.com/2014/09/20/spotify-engineering-culture-part-2/

6 Learn how a Computer Works

6.1 CPU,RAM,GPU,HDD

6.2 Differences between PCs and Servers

I talked about computer hardware and GPU processing in this podcast: https://anchor.fm/andreaskayy/embed/episodes/Why-the-hardware-and-the-GPU-is-super-important--PoDS-030-e23

7 Computer Networking - Data Transmission

7.1 OSI Model

The OSI Model describes how data is flowing through the network. It consists of layers starting from physical layers, basically how the data is transmitted over the line or optic fiber.

Cisco page that shows the layers of the OSI model and how it works: https://learningnetwork.cisco.com/docs/DOC-30382

Check out this page: https://www.studytonight.com/computer-networks/complete-osi-model

The Wikipedia page is also very good: https://en.wikipedia.org/wiki/OSI_model

Which protocol lives on which layer? Check out this network protocol map. Unfortunately it is really hard to find it theses days: https://www.blackmagicboxes.com/wp-content/uploads/2016/12/Network-Protocols-Map-Poster.jpg

7.2 IP Subnetting

Check out this IP Adress and Subnet guide from Cisco: https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html

A calculator for Subnets: https://www.calculator.net/ip-subnet-calculator.html

7.3 Switch, Level 3 Switch

7.4 Router

7.5 Firewalls

I talked about Network Infrastructure and Techniques in this podcast: https://anchor.fm/andreaskayy/embed/episodes/IT-Networking-Infrastructure-and-Linux-031-PoDS-e242bh

8 Security and Privacy

8.1 SSL Public & Private Key Certificates

8.2 What is a certificate authority

8.3 JSON Web Tokens

Link to the Wiki page: https://en.wikipedia.org/wiki/JSON_Web_Token

8.4 GDPR regulations

The EU created the GDPR "General Data Protection Regulation" to protect your personal data like: Your name, age, where you live and so on.

It's huge and quite complicated. If you want to do online business in the EU you need to apply these rules. The GDPR is applicable since May 25th 2018. So, if you haven't looked into it, now is the time.

The penalties can be crazy high if you do mistakes here.

Check out the full GDPR regulation here: https://gdpr-info.eu

By the way, if you do profiling or in general analyse big data, look into it. There are some important regulations. Unfortunately.

I spend months with GDPR compliance. Super fun. Not! Hahaha

8.5 Privacy by design

When should you look into privacy regulations and solutions?

Creating the product or service first and then bolting on the privacy is a bad choice. The best way is to start implementing privacy right away in the engineering phase.

This is called privacy by design. Privacy as an integral part of your business, not just something optional.

Check out the Wikipedia page to get a feeling of the important principles: https://en.wikipedia.org/wiki/Privacy_by_design

9 Linux

Linux is very important to learn, at least the basics. Most Big Data tools or NoSQL databases are running on Linux.

From time to time you need to modify stuff through the operation system. Especially if you run an infrastructure as a service solution like Cloudera CDH, Hortonworks or a MapR Hadoop distribution.

9.1 OS Basics

Show all historic commands

```
history | grep docker
```

9.2 Shell scripting

Äh, creating shell scripts in 2019? Believe it or not scripting in the command line is still important.

Start a process, automatically rename, move or do a quick compaction of log files. It still makes a lot of sense.

Check out this cheat sheet to get started with scripting in Linux: https://devhints.io/bash

There's also this Medium article with a super simple example for beginners: https://medium.com/@saswat.sipun/shell-scripting-cheat-sheet-c0ecfb80391

9.3 Cron jobs

Cron jobs are super important to automate simple processes or jobs in Linux. You need this here and there I promise. Check out this three guides:

https://linuxconfig.org/linux-crontab-reference-guide

https://www.ostechnix.com/a-beginners-guide-to-cron-jobs/

And of course Wikipedia, which is surprisingly good: https://en.wikipedia.org/wiki/Cron

Pro tip: Don't forget to end your cron files with an empty line or a comment, otherwise it will not work.

9.4 Packet management

Linux Tips are the second part of this podcast: https://anchor.fm/andreaskayy/embed/episodes/IT-Networking-Infrastructure-and-Linux-031-PoDS-e242bh

10 The Cloud

10.1 IaaS vs PaaS vs SaaS

Check out this Podcast it will help you understand where's the difference and how to decide on what you are going to use.

Podcast Episode: #082 Reading Tweets With Apache Nifi & IaaS vs PaaS vs SaaS	
In this episode we are talking about the differences between infrastructure as a service, platform as a service and application as a service. Then we install the Nifi docker container and look into how we can extract the twitter data.	
Youtube	Click here to watch
Audio	Click here to listen

Table 10.1: Podcast: 082 Reading Tweets With Apache Nifi & IaaS vs PaaS vs SaaS

10.2 AWS, Azure, IBM, Google

Each of these have there own answer to IaaS, Paas and SaaS. Pricing and pricing models vary greatly between each provider. Likewise each provider's service may have limitations and strengths.

10.2.1 AWS

Full list of AWS services.

10.2.2 Azure

10.2.3 IBM

10.2.4 Google

Google's offerings referred to as Google Cloud Platform provides wide variety of services that is ever evolving. List of GCP services with brief description. In recent years documentation and tutorials have com a long way to help getting started with GCP. You can start with a free account but to use many of the services you will need to turn on billing. Once you do enable billing always remember to turn off services that you have spun up for learning purposes. It is also a good idea to turn on billing limits and alerts.

10.3 Cloud vs On-Premises

Podcast Episode: #076 Cloud vs On-Premise	
How do you choose between Cloud vs On-Premises, pros and cons and what you have to think about. Because there are good reasons to not go cloud. Also thoughts on how to choose between the cloud providers by just comparing instance prices. Otherwise the comparison will drive you insane. My suggestion: Basically use them as IaaS and something like Cloudera as PaaS. Then build your solution on top of that.	
YouTube	Click here to watch
Audio	Click here to listen

Table 10.2: Podcast: 076 Cloud vs On-Premise

10.4 Security

Listen to a few thoughts about the cloud in this podcast: https://anchor.fm/andreaskayy/embed/episodes/Dont-Be-Arrogant-The-Cloud-is-Safer-Then-Your-On-Premise-e16k9s

10.5 Hybrid Clouds

Hybrid clouds are a mixture of on-premises and cloud deployment. A very interesting example for this is Google Anthos:

https://cloud.google.com/anthos/

11 Security Zone Design

11.1 How to secure a multi layered application

(UI in different zone then SQL DB)

11.2 Cluster security with Kerberos

I talked about security zone design and lambda architecture in this podcast: https://anchor.fm/andreaskayy/embed/episodes/How-to-Design-Security-Zones-and-Lambda-Architecture--P

12 Big Data

12.1 What is big data and where is the difference to data science and data analytics?

I talked about the difference in this podcast: https://anchor.fm/andreaskayy/embed/episodes/BI-vs-Data-Science-vs-Big-Data-e199hq

12.2 The 4 Vs of Big Data

It is a complete misconception. Volume is only one part of the often called four V's of big data: Volume, velocity, variety and veracity.

Volume is about the size - How much data you have

Velocity is about the speed - How fast data is getting to you

How much data in a specific time needs to get processed or is coming into the system. This is where the whole concept of streaming data and real-time processing comes in to play.

Variety is about the variety - How different your data is

Like CSV files, PDFs that you have and stuff in XML. That you also have JSON logfiles, or data in some kind of a key-value store.

It's about the variety of data types from different sources that you basically want to join together. All to make an analysis based on that data.

Veracity is about the credibility - How reliable your data is

The issue with big data is, that it is very unreliable.

You cannot really trust the data. Especially when you're coming from the Internet of Things (IoT) side. Devices use sensors for measurement of temperature, pressure,

acceleration and so on.

You cannot always be hundred percent sure that the actual measurement is right.

When you have data that is from for instance SAP and it contains data that is created by hand you also have problems. As you know we humans are bad at inputting stuff.

Everybody articulates different. We make mistakes, down to the spelling and that can be a very difficult issue for analytics.

I talked about the 4Vs in this podcast: https://anchor.fm/andreaskayy/embed/episodes/4-Vs-Of-Big-Data-Are-Enough-e1h2ra

12.3 Why Big Data?

What I always emphasize is the four V's are quite nice. They give you a general direction.

There is a much more important issue: Catastrophic Success.

What I mean by catastrophic success is, that your project, your startup or your platform has more growth that you anticipated. Exponential growth is what everybody is looking for.

Because with exponential growth there is the money. It starts small and gets very big very fast. The classic hockey stick curve:

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, ... BOOM!

Think about it. It starts small and quite slow, but gets very big very fast.

You get a lot of users or customers who are paying money to use your service, the platform or whatever. If you have a system that is not equipped to scale and process the data the whole system breaks down.

That's catastrophic success. You are so successful and grow so fast that you cannot fulfill the demand anymore. And so you fail and it's all over.

It's now like you just can make that up while you go. That you can foresee in a few months or weeks the current system doesn't work anymore.

12.3.1 Planning is Everything

It's all happens very very fast and you cannot react anymore. There's a necessary type of planning and analyzing the potential of your business case necessary.

Then you need to decide if you actually have big data or not.

You need to decide if you use big data tools. This means when you conceptualize the whole infrastructure it might look ridiculous to actually focus on big data tools.

But in the long run it will help you a lot. Good planning will get a lot of problems out of the way, especially if you think about streaming data and real-time analytics.

12.3.2 The problem with ETL

A typical old-school platform deployment would look like the picture below. Devices use a data API to upload data that gets stored in a SQL database. An external analytics tool is querying data and uploading the results back to the SQL DB. Users then use the user interface to display data stored in the database.

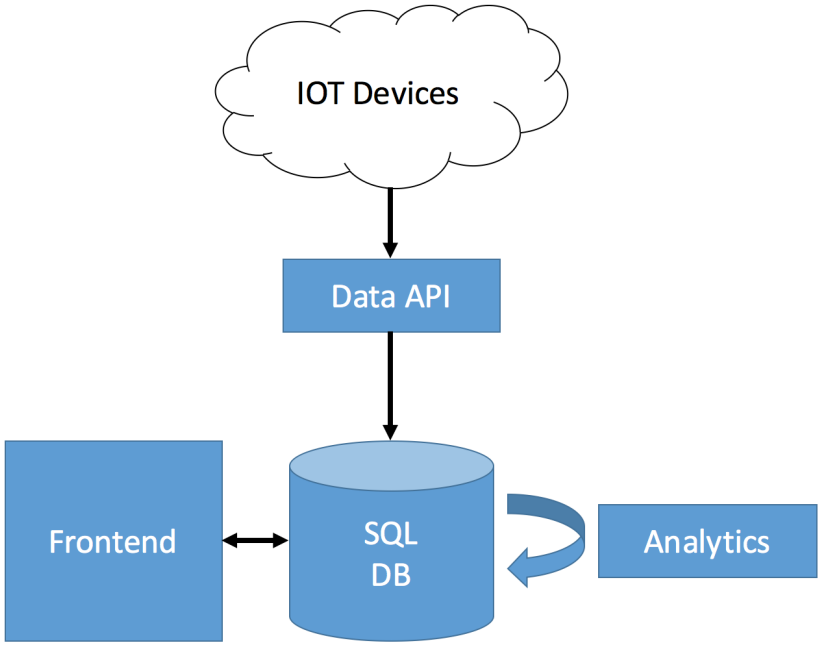

Figure 12.1: Common SQL Platform Architecture

Now, when the front end queries data from the SQL database the following three steps happen:

- The database extracts all the needed rows from the storage. (E) - The extracted data

gets transformed, for instance sorted by timestamp or something a lot more complex. (T) - The transformed data is loaded to the destination (the user interface) for chart creation. (L)

With exploding amounts of stored data the ETL process starts being a real problem.

Analytics is working with large data sets, for instance whole days, weeks, months or more. Data sets are very big like 100GB or Terabytes. That means Billions or Trillions of rows.

This has the result that the ETL process for large data sets takes longer and longer. Very quickly the ETL performance gets so bad it won't deliver results to analytics anymore.

A traditional solution to overcome these performance issues is trying to increase the performance of the database server. That's what's called scaling up.

12.3.3 Scaling Up

To scale up the system and therefore increase ETL speeds administrators resort to more powerful hardware by:

Speeding up the extract performance by adding faster disks to physically read the data faster. Increasing RAM for row caching. What is already in memory does not have to be read by slow disk drives. Using more powerful CPU's for better transform performance (more RAM helps here as well). Increasing or optimising networking performance for faster data delivery to the front end and analytics.

In summary: Scaling up the system is fairly easy.

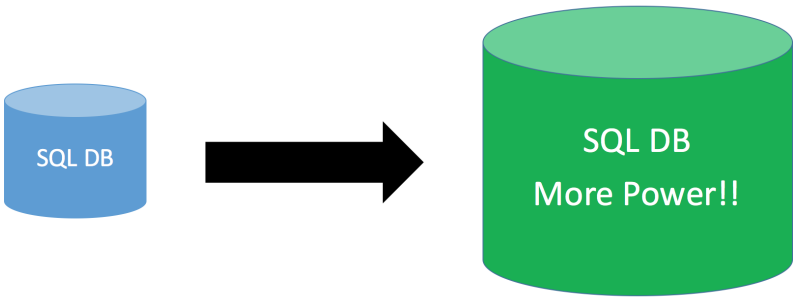

Figure 12.2: Scaling up a SQL Database

But with exponential growth it is obvious that sooner or later (more sooner than later) you will run into the same problems again. At some point you simply cannot scale up anymore because you already have a monster system, or you cannot afford to buy more expensive hardware.

The next step you could take would be scaling out.

12.3.4 Scaling Out

Scaling out is the opposite of scaling up. Instead of building bigger systems the goal is to distribute the load between many smaller systems.

The easiest way of scaling out an SQL database is using a storage area network (SAN) to store the data. You can then use up to eight SQL servers (explain), attach them to the SAN and let them handle queries. This way load gets distributed between those eight servers.

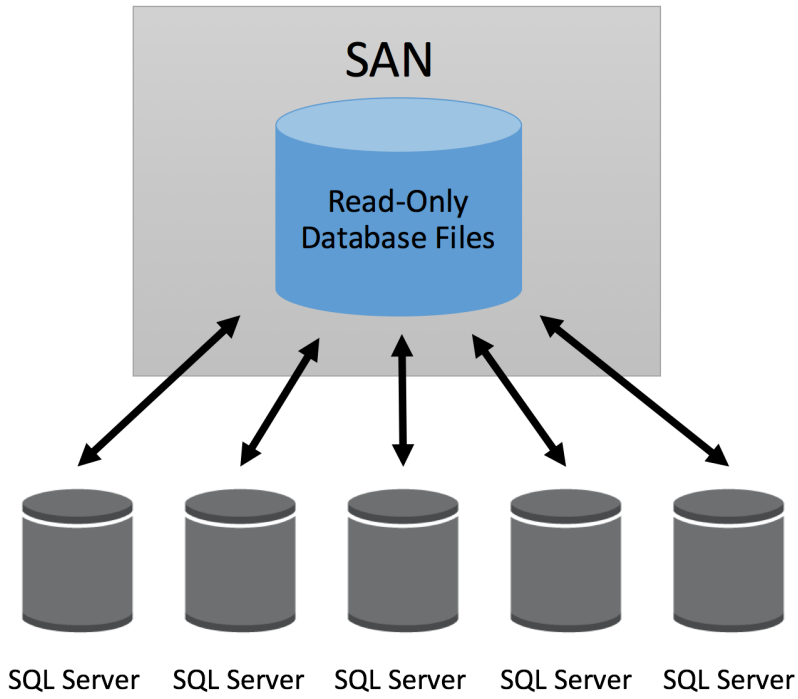

Figure 12.3: Scaling out a SQL Database

One major downside of this setup is that, because the storage is shared between the SQL servers, it can only be used as an read only database. Updates have to be done periodically, for instance once a day. To do updates all SQL servers have to detach from the database. Then, one is attaching the DB in read-write mode and refreshing the data. This procedure can take a while if a lot of data needs to be uploaded.

This Link (missing) to a Microsoft MSDN page has more options of scaling out an SQL database for you.

I deliberately don't want to get into details about possible scaling out solutions. The

point I am trying to make is that while it is possible to scale out SQL databases it is very complicated.

There is no perfect solution. Every option has its up- and downsides. One common major issue is the administrative effort that you need to take to implement and maintain a scaled out solution.

12.3.5 Please don't go Big Data

If you don't run into scaling issues please, do not use big data tools!

Big data is a expensive thing. A Hadoop cluster for instance needs at least five servers to work properly. More is better.

Believe me this stuff costs a lot of money.

Especially when you are talking about maintenance and development on top big data tools into account.

If you don't need it it's making absolutely no sense at all!

On the other side: If you really need big data tools they will save your ass :)

13 My Big Data Platform Blueprint

Some time ago I have created a simple and modular big data platform blueprint for myself. It is based on what I have seen in the field and read in tech blogs all over the internet.

Today I am going to share it with you.

Why do I believe it will be super useful to you?

Because, unlike other blueprints it is not focused on technology. It is based on four common big data platform design patterns.

Following my blueprint will allow you to create the big data platform that fits exactly your needs. Building the perfect platform will allow data scientists to discover new insights.

It will enable you to perfectly handle big data and allow you to make data driven decisions.

THE BLUEPRINT The blueprint is focused on the four key areas: Ingest, store, analyse and display.

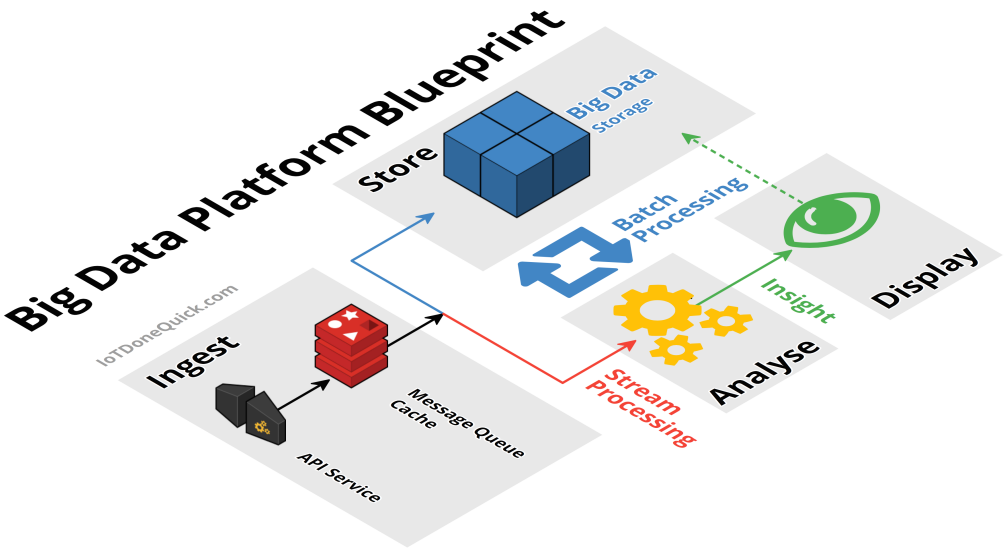

Figure 13.1: Platform Blueprint

Having the platform split like this turns it it a modular platform with loosely coupled interfaces.

Why is it so important to have a modular platform?

If you have a platform that is not modular you end up with something that is fixed or hard to modify. This means you can not adjust the platform to changing requirements of the company.

Because of modularity it is possible to replace every component, if you need it.

Now, lets talk more about each key area.

13.1 Ingest

Ingestion is all about getting the data in from the source and making it available to later stages. Sources can be everything from tweets, server logs to IoT sensor data (e.g. from cars).

Sources send data to your API Services. The API is going to push the data into a temporary storage.

The temporary storage allows other stages simple and fast access to incoming data.

A great solution is to use messaging queue systems like Apache Kafka, RabbitMQ or AWS Kinesis. Sometimes people also use caches for specialised applications like Redis.

A good practice is that the temporary storage follows the publish-subscribe pattern. This way APIs can publish messages and Analytics can quickly consume them.

13.2 Analyse / Process

The analyse stage is where the actual analytics is done. Analytics, in the form of stream and batch processing.

Streaming data is taken from ingest and fed into analytics. Streaming analyses the “live” data, thus generating fast results.

As the central and most important stage, analytics also has access to the big data storage. Because of that connection, analytics can take a big chunk of data and analyse it.

This type of analysis is called batch processing. It will deliver you answers for the big questions.

To learn more about stream and batch processing read my blog post: missing How to Create New and Exciting Big Data Aided Products

The analytics process, batch or streaming, is not a one way process. Analytics also can write data back to the big data storage.

Often times writing data back to the storage makes sense. It allows you to combine previous analytics outputs with the raw data.

Analytics insight can give meaning to the raw data when you combine them. This combination will often times allow you to create even more useful insight.

A wide variety of analytics tools are available. Ranging from MapReduce or AWS Elastic MapReduce to Apache Spark and AWS lambda.

13.3 Store

This is the typical big data storage where you just store everything. It enables you to analyse the big picture.

Most of the data might seem useless for now, but it is of upmost importance to keep it. Throwing data away is a big no no.

Why not throw something away when it is useless?

Although it seems useless for now, data scientists can work with the data. They might find new ways to analyse the data and generate valuable insight from it.

What kind of systems can be used to store big data?

Systems like Hadoop HDFS, Hbase, Amazon S3 or DynamoDB are a perfect fit to store big data.

Check out my podcast how to decide between SQL and NoSQL: https://anchor.fm/andreaskayy/embed/episodes/NoSQL-Vs-SQL-How-To-Choose-e12f1o

13.4 Display

Displaying data is as important as ingesting, storing and analysing it. People need to be able to make data driven decisions.

This is why it is important to have a good visual presentation of the data. Sometimes you have a lot of different use cases or projects using the platform.

It might not be possible for you to build the perfect UI that fits everyone. What you should do in this case is enable others to build the perfect UI themselves.

How to do that? By creating APIs to access the data and making them available to developers.

Either way, UI or API the trick is to give the display stage direct access to the data in the big data cluster. This kind of access will allow the developers to use analytics results as well as raw data to build the the perfect application.

14 Lambda Architecture

Podcast Episode: #077 Lambda Architecture and Kappa Architecture	
In this stream we talk about the lambda architecture with stream and batch processing as well as a alternative the Kappa Architecture that consists only of streaming. Also Data engineer vs data scientist and we discuss Andrew Ng's AI Transformation Playbook	
Audio	Click here to listen
Youtube	Click here to watch

Table 14.1: Podcast: 077 Lambda Architecture and Kappa Architecture

14.1 Batch Processing

Ask the big questions. Remember your last yearly tax statement?

You break out the folders. You run around the house searching for the receipts.

All that fun stuff.

When you finally found everything you fill out the form and send it on its way.

Doing the tax statement is a prime example of a batch process.

Data comes in and gets stored, analytics loads the data from storage and creates an output (insight):

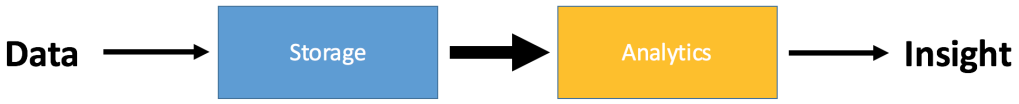

Figure 14.1: Batch Processing Pipeline

Batch processing is something you do either without a schedule or on a schedule (tax statement). It is used to ask the big questions and gain the insights by looking at the big picture.

To do so, batch processing jobs use large amounts of data. This data is provided by storage systems like Hadoop HDFS.

They can store lots of data (petabytes) without a problem.

Results from batch jobs are very useful, but the execution time is high. Because the amount of used data is high.

It can take minutes or sometimes hours until you get your results.

14.2 Stream Processing

Gain instant insight into your data.

Streaming allows users to make quick decisions and take actions based on "real-time" insight. Contrary to batch processing, streaming processes data on the fly, as it comes in.

With streaming you don't have to wait minutes or hours to get results. You gain instant insight into your data.

In the batch processing pipeline, the analytics was after the data storage. It had access to all the available data.

Stream processing creates insight before the data storage. It has only access to fragments of data as it comes in.

As a result the scope of the produced insight is also limited. Because the big picture is missing.

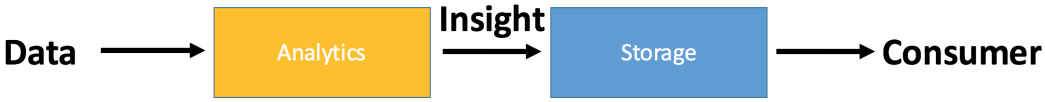

Figure 14.2: Stream Processing Pipeline

Only with streaming analytics you are able to create advanced services for the customer. Netflix for instance incorporated stream processing into Chuckwa V2.0 and the new Keystone pipeline.

One example of advanced services through stream processing is the Netflix "Trending Now" feature. Check out the Netflix case study.

14.3 Should you do stream or batch processing?

It is a good idea to start with batch processing. Batch processing is the foundation of every good big data platform.

A batch processing architecture is simple, and therefore quick to set up. Platform simplicity means, it will also be relatively cheap to run.

A batch processing platform will enable you to quickly ask the big questions. They will give you invaluable insight into your data and customers.

When the time comes and you also need to do analytics on the fly, then add a streaming pipeline to your batch processing big data platform.

14.4 Lambda Architecture Alternative

14.4.1 Kappa Architecture

14.4.2 Kappa Architecture with Kudu

14.5 Why a Good Data Platform Is Important

Podcast Episode: #066 How To Do Data Science From A Data Engineers Perspective	
A simple introduction how to do data science in the context of the internet of things.	
YouTube	Click here to watch
Audio	Click here to listen

Table 14.2: Podcast: 066 How To Do Data Science From A Data Engineers..

15 Data Warehouse vs Data Lake

Podcast Episode: #055 Data Warehouse vs Data Lake	
On this podcast we are going to talk about data warehouses and data lakes? When do people use which? What are the pros and cons of both? Architecture examples for both Does it make sense to completely move to a data lake?	
YouTube	Click here to watch
Audio	Click here to listen

Table 15.1: Podcast: 055 Data Warehouse vs Data Lake

16 Hadoop Platforms

When people talk about big data, one of the first things come to mind is Hadoop. Google's search for Hadoop returns about 28 million results.

It seems like you need Hadoop to do big data. Today I am going to shed light onto why Hadoop is so trendy.

You will see that Hadoop has evolved from a platform into an ecosystem. Its design allows a lot of Apache projects and 3rd party tools to benefit from Hadoop.

I will conclude with my opinion on, if you need to learn Hadoop and if Hadoop is the right technology for everybody.

16.1 What is Hadoop

Hadoop is a platform for distributed storing and analyzing of very large data sets.

Hadoop has four main modules: Hadoop common, HDFS, MapReduce and YARN. The way these modules are woven together is what makes Hadoop so successful.

The Hadoop common libraries and functions are working in the background. That's why I will not go further into them. They are mainly there to support Hadoop's modules.

Podcast Episode: #060 What Is Hadoop And Is Hadoop Still Relevant In 2019?	
An introduction into Hadoop HDFS, YARN and MapReduce. Yes, Hadoop is still relevant in 2019 even if you look into serverless tools.	
YouTube	Click here to watch
Audio	Click here to listen

Table 16.1: Podcast: 060 What Is Hadoop And Is Hadoop Still Relevant In 2019?

16.2 What makes Hadoop so popular?

Storing and analyzing data as large as you want is nice. But what makes Hadoop so popular?

Hadoop's core functionality is the driver of Hadoop's adoption. Many Apache side projects use it's core functions.

Because of all those side projects Hadoop has turned more into an ecosystem. An ecosystem for storing and processing big data.

To better visualize this eco system I have drawn you the following graphic. It shows some projects of the Hadoop ecosystem who are closely connected with the Hadoop.

It is not a complete list. There are many more tools that even I don't know. Maybe I am drawing a complete map in the future.

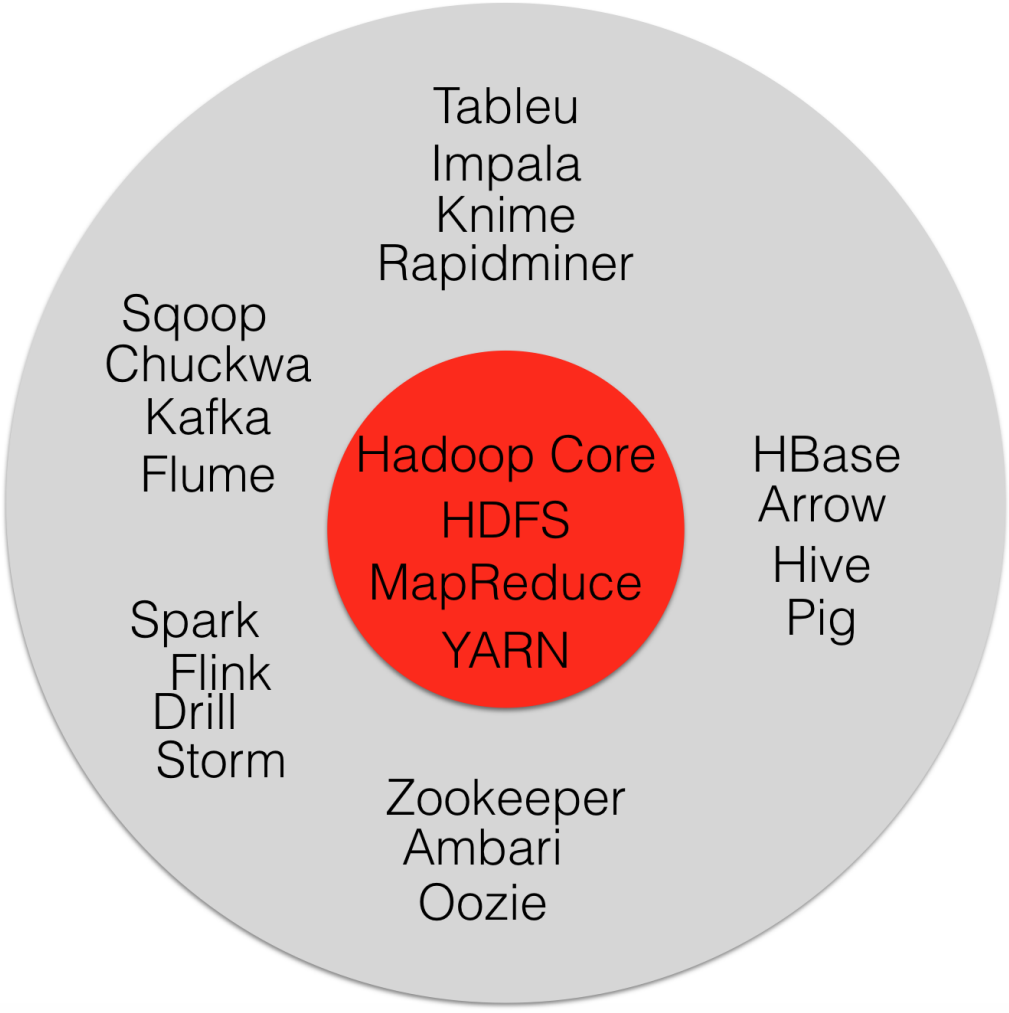

Figure 16.1: Hadoop Ecosystem Components

16.3 Hadoop Ecosystem Components

Remember my big data platform blueprint? The blueprint has four stages: Ingest, store, analyse and display.

Because of the Hadoop ecosystem the different tools in these stages can work together perfectly.

Here's an example:

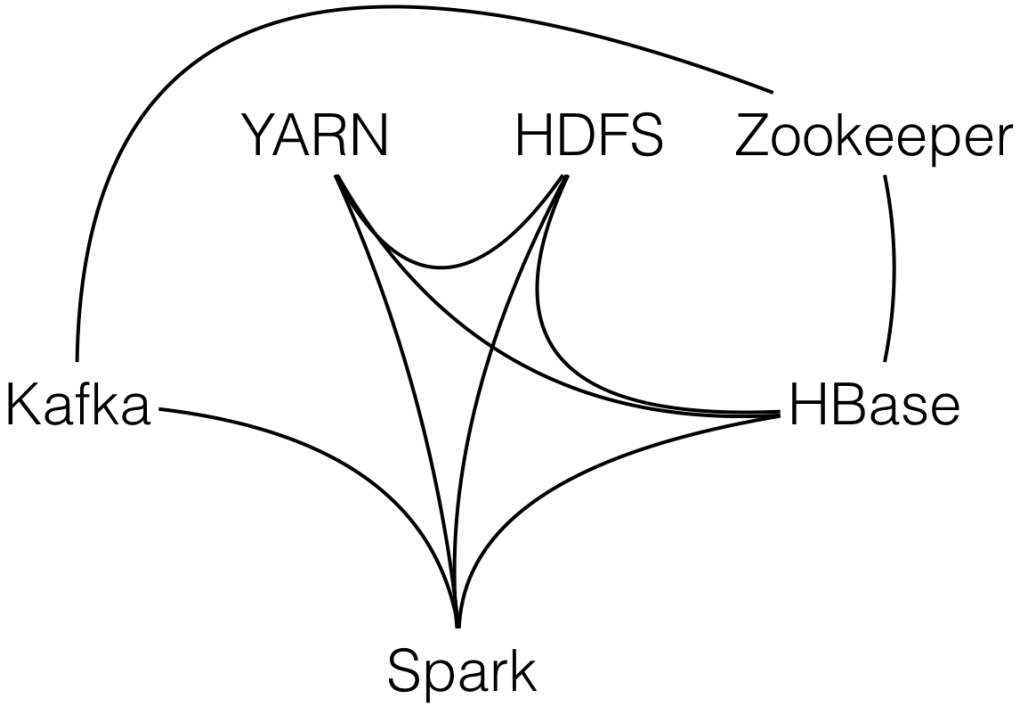

Figure 16.2: Connections between tools

You use Apache Kafka to ingest data, and store the it in HDFS. You do the analytics with Apache Spark and as a backend for the display you store data in Apache HBase.

To have a working system you also need YARN for resource management. You also need Zookeeper, a configuration management service to use Kafka and HBase

As you can see in the picture below each project is closely connected to the other.

Spark for instance, can directly access Kafka to consume messages. It is able to access HDFS for storing or processing stored data.

It also can write into HBase to push analytics results to the front end.

The cool thing of such ecosystem is that it is easy to build in new functions.

Want to store data from Kafka directly into HDFS without using Spark?

No problem, there is a project for that. Apache Flume has interfaces for Kafka and HDFS.

It can act as an agent to consume messages from Kafka and store them into HDFS. You even do not have to worry about Flume resource management.

Flume can use Hadoop's YARN resource manager out of the box.

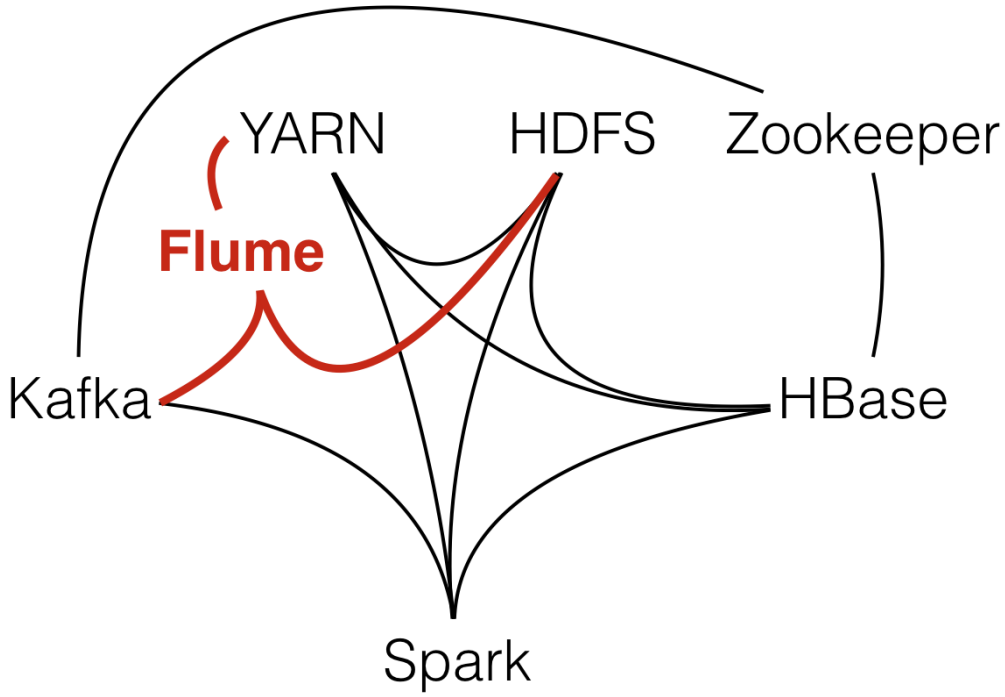

Figure 16.3: Flume Integration

16.4 Hadoop Is Everywhere?

Although Hadoop is so popular it is not the silver bullet. It isn't the tool that you should use for everything.

Often times it does not make sense to deploy a Hadoop cluster, because it can be overkill. Hadoop does not run on a single server.

You basically need at least five servers, better six to run a small cluster. Because of that. the initial platform costs are quite high.

One option you have is to use a specialized systems like Cassandra, MongoDB or other NoSQL DB's for storage. Or you move to Amazon and use Amazon's Simple Storage Service, or S3.

Guess what the tech behind S3 is. Yes, HDFS. That's why AWS also has the equivalent to MapReduce named Elastic MapReduce.

The great thing about S3 is that you can start very small. When your system grows you don't have to worry about S3's server scaling.

16.5 Should you learn Hadoop?

Yes, I definitely recommend you to get to know how Hadoop works and how to use it. As I have shown you in this article, the ecosystem is quite large.

Many big data projects use Hadoop or can interface with it. That's why it is generally a good idea to know as many big data technologies as possible.

Not in depth, but to the point that you know how they work and how you can use them. Your main goal should be to be able to hit the ground running when you join a big data project.

Plus, most of the technologies are open source. You can try them out for free.

16.6 How does a Hadoop System architecture look like

16.7 What tools are usually in a with Hadoop Cluster

Yarn Zookeeper HDFS Oozie Flume Hive

16.8 How to select Hadoop Cluster Hardware

17 Docker

17.1 What is docker and what do you use it for

Have you played around with Docker yet? If you're a data science learner or a data scientist you need to check it out!

It's awesome because it simplifies the way you can set up development environments for data science. If you want to set up a dev environment you usually have to install a lot of packages and tools.

17.1.1 Don't Mess Up Your System

What this does is you basically mess up your operating system. If you're a starter you don't know which packages you need to install. You don't know which tools you need to install.

If you want to for instance start with Jupyter notebooks you need to install that on your PC somehow. Or you need to start installing tools like PyCharm or Anaconda.

All that gets added to your system and so you mess up your system more and more and more. What Docker brings you, especially if you're on a Mac or a Linux system is simplicity.

17.1.2 Preconfigured Images

Because it is so easy to install on those systems. Another cool thing about docker images is you can just search them in the Docker store, download them and install them on your system.

Running them in a completely pre-configured environment. You don't need to think about stuff, you go to the Docker library you search for Deep Learning, GPU and Python.

You get a list of images you can download. You download one, start it up, you go to the browser hit up the URL and just start coding.

Start doing the work. The only other thing you need to do is bind some drives to that instance so you can exchange files. And then that's it!

There is no way that you can crash or mess up your system. It's all encapsulated into Docker. Why this works is because Docker has natively access to your hardware.

17.1.3 Take It With You

It's not a completely virtualized environment like a VirtualBox. An image has the upside that you can take it wherever you want. So if you're on your PC at home use that there.

Make a quick build, take the image and go somewhere else. Install the image which is usually quite fast and just use it like you're at home.

It's that awesome!

17.2 Kubernetes Container Deployment

I am getting into Docker a lot more myself. For a bit different reasons.

What I'm looking for is using Docker with Kubernetes. With Kubernetes you can automate the whole container deployment process.

The idea with is that you have a cluster of machines. Lets say you have a 10 server cluster and you run Kubernetes on it.

Kubernetes lets you spin up Docker containers on-demand to execute tasks. You can set up how much resources like CPU, RAM, Network, Docker container can use.

You can basically spin up containers, on the cluster on demand. When ever you need to do a analytics task.

Perfect for Data Science.

17.3 How to create, start, stop a Container

17.4 Docker micro services?

17.5 Kubernetes

17.6 Why and how to do Docker container orchestration

Podcast about how data science learners use Docker (for data scientists): https://anchor.fm/andreaskayy/embed/episodes/Learn-Data-Science-Go-Docker-e10n7u

17.7 Useful Docker Commands

Create a container:

```
docker run CONTAINER --network NETWORK
```

Start a stopped container:

```
docker start CONTAINER NAME
```

Stop a running container:

```
docker stop
```

List all running containers

```
docker ps
```

List all containers including stopped ones

```
docker ps -a
```

Inspect the container configuration. For instance network settings and so on:

```
docker inspect CONTAINER
```

List all available virtual networks:

```
docker network ls
```

Create a new network:

```
docker network create NETWORK --driver bridge
```

Connect a running container to a network

```
docker network connect NETWORK CONTAINER
```

Disconnect a running container from a network

```
docker network disconnect NETWORK CONTAINER
```

Remove a network

```
docker network rm NETWORK
```

18 REST APIs

APIs or Application Programming Interfaces are the cornerstones of any great data platform.

Podcast Episode: #033 How APIs Rule The World	
Strong APIs make a good platform. In this episode I talk about why you need APIs and why Twitter is a great example. Especially JSON APIs are my personal favorite. Because JSON is also important in the Big Data world, for instance in log analytics. How? Check out this episode!	
Audio	Click here to listen

Table 18.1: Podcast: 033 How APIs Rule The World

18.1 API Design

In this podcast episode we look into the Twitter API. It's a great example how to build an API

Podcast Episode: #081 Twitter API Research Data Engineering Course Part 5	
In this episode we look into the Twitter API documentation, which I love by the way. How can we get old tweets for a certain hashtags and how to get current live tweets for these hashtags?	
Audio	Click here to listen
Youtube	Click here to watch

Table 18.2: Podcast: 081 Twitter API Research

18.2 Implementation Frameworks

Jersey:

https://jersey.github.io/documentation/latest/getting-started.html

Swagger:

https://github.com/swagger-api/swagger-core/wiki/Swagger-2.X---Getting-started

Jersey vs Swagger:

https://stackoverflow.com/questions/36997865/what-is-the-difference-between-swagger-api-and-jax-rs

Spring Framework:

https://spring.io/

When to use Spring or Jersey:

https://stackoverflow.com/questions/26824423/what-is-the-difference-among-spring-rest-service-and-j

18.3 OAuth security

19 Databases

19.1 SQL Databases

19.1.1 PostgreSQL DB

Homepage:

https://www.postgresql.org/

PostgreSQL vs MongoDB:

https://blog.panoply.io/postgresql-vs-mongodb

19.1.2 Database Design

19.1.3 SQL Queries

19.1.4 Stored Procedures

19.1.5 ODBC/JDBC Server Connections

19.2 NoSQL Stores

19.2.1 KeyValue Stores (HBase)

19.2.2 Document Store HDFS

The Hadoop distributed file system, or HDFS, allows you to store files in Hadoop. The difference between HDFS and other file systems like NTFS or EXT is that it is a distributed one.

Podcast Episode: #056 NoSQL Key Value Stores Explained with HBase	
What is the difference between SQL and NoSQL? In this episode I show you on the example of HBase how a key/value store works.	
YouTube	Click here to watch
Audio	Click here to listen

Table 19.1: Podcast: 056 NoSQL Key Value Stores Explained with HBase

What does that mean exactly?

A typical file system stores your data on the actual hard drive. It is hardware dependent.

If you have two disks then you need to format every disk with its own file system. They are completely separate.

You then decide on which disk you physically store your data.

HDFS works different to a typical file system. HDFS is hardware independent.

Not only does it span over many disks in a server. It also spans over many servers.

HDFS will automatically place your files somewhere in the Hadoop server collective.

It will not only store your file, Hadoop will also replicate it two or three times (you can define that). Replication means replicas of the file will be distributed to different servers.

This gives you superior fault tolerance. If one server goes down, then your data stays available on a different server.

Another great thing about HDFS is, that there is no limit how big the files can be. You can have server log files that are terabytes big.

How can files get so big? HDFS allows you to append data to files. Therefore, you can continuously dump data into a single file without worries.

HDFS physically stores files different then a normal file system. It splits the file into blocks.

These blocks are then distributed and replicated on the Hadoop cluster. The splitting happens automatically.

In the configuration you can define how big the blocks should be. 128 megabyte or 1 gigabyte?

No problem at all.

This mechanic of splitting a large file in blocks and distributing them over the servers is great for processing. See the MapReduce section for an example.

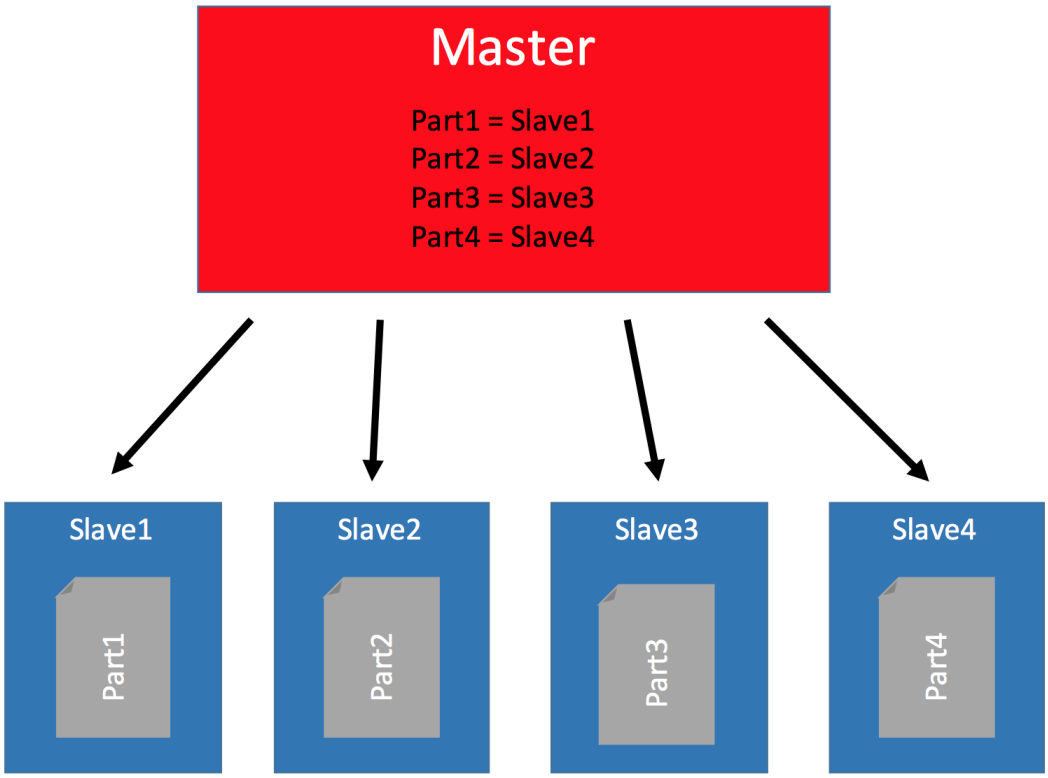

Figure 19.1: HDFS Master and Data Nodes

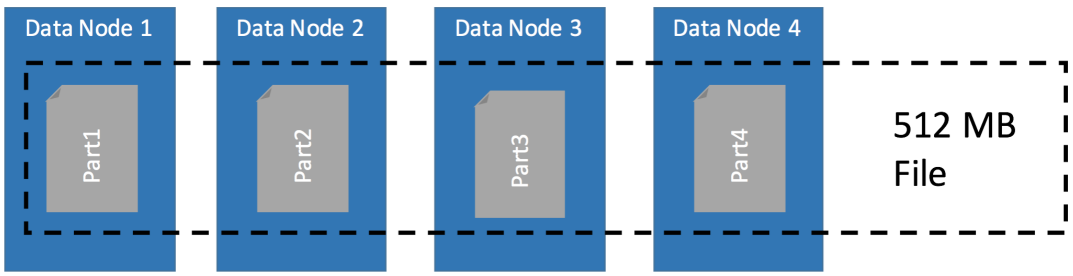

Figure 19.2: Distribution of Blocks for a 512MB File

19.2.3 Document Store MongoDB

Podcast Episode: #093 What is MongoDB	
I was always curious about MongoDB. In this stream we go over the links you can find below	
YouTube	Click here to watch

Table 19.2: Podcast: 093 What is MongoDB

Links:

What is MongoDB:

https://www.guru99.com/what-is-mongodb.html#4

Or directly from MongoDB.com:

https://www.mongodb.com/what-is-mongodb

Storage in BSON files:

https://en.wikipedia.org/wiki/BSON

Hello World in MongoDB:

https://www.mkyong.com/mongodb/mongodb-hello-world-example

Real-Time Analytics on MongoDB Data in Power BI:

https://dzone.com/articles/real-time-analytics-on-mongodb-data-in-power-bi

Spark and MongoDB:

https://www.mongodb.com/scale/when-to-use-apache-spark-with-mongodb

MongoDB vs Time Series Database:

https://blog.timescale.com/how-to-store-time-series-data-mongodb-vs-timescaledb-postgresql-a739397

Fun article titled why you should never use mongodb:

http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/

MongoDB vs Cassandra:

https://blog.panoply.io/cassandra-vs-mongodb

19.2.4 Elasticsearch Search Engine and Document Store

Elasticsearch is not a DB but firstly a search engine that indexes JSON documents.

Podcast Episode: #095 What is Elasticsearch & Why is It So Popular?	
Elasticsearch is a super popular tool for indexing and searching data. On this stream we check out how it works, architectures and what to use it for. There must be a reason why it is so popular.	
YouTube	Click here to watch

Table 19.3: Podcast: What is Elasticsearch & Why is It So Popular?

Links:

Great example for architecture with Elasticsearch, Logstash and Kibana:
https://www.elastic.co/pdf/architecture-best-practices.pdf

Introduction to Elasticsearch in the documentation:
https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html

Working with JSON documents:
https://www.slideshare.net/openthinklabs/03-elasticsearch-data-in-data-out

JSONs need to be flattened heres how to work with nested objects in the JSON:
https://www.elastic.co/guide/en/elasticsearch/reference/current/nested.html

Indexing basics:
https://www.slideshare.net/knoldus/deep-dive-into-elasticsearch

How to query data with DSL language:
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-common-terms-query.html

How to do searches with search API:
https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html

General recommendations when working with Elasticsearch:
https://www.elastic.co/guide/en/elasticsearch/reference/current/general-recommendations.html

JSON document example and intro to Kibana:
https://www.slideshare.net/objectrocket/an-intro-to-elasticsearch-and-kibana

How to connect Tableau to Elasticsearch:
https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-client-apps-tableau.html

Benchmarks how fast Elasticsearch is:
https://medium.appbase.io/benchmarking-elasticsearch-1-million-writes-per-sec-bf37e7ca8a4c

Elasticsearch vs MongoDB quick overview:
https://db-engines.com/en/system/Elasticsearch%3BMongoDB

Logstash overview (preprocesses data before insert into Elasticsearch) https://www.elastic.co/products/logstash

X-Pack Security for Elasticsearch:
https://www.elastic.co/guide/en/elasticsearch/reference/current/security-api.html

Google Trends Grafana vs Kibana:
https://trends.google.com/trends/explore?geo=US&q=%2Fg%2F11fy132gmf,%2Fg%2F11cknd0blr

19.2.5 Hive Warehouse

19.2.6 Impala

19.2.7 Kudu

19.2.8 Apache Druid

Podcast Episode: Druid NoSQL DB and Analytics DB Introduction	
In this video I explain what Druid is and how it works. We look into the architecture of a Druid cluster and check out how Clients access the data.	
YouTube	Click here to watch

Table 19.4: Podcast: Druid NoSQL DB and Analytics DB Introduction

19.2.9 InfluxDB Time Series Database

Key concepts:

https://docs.influxdata.com/influxdb/v1.7/concepts/key_concepts/

InfluxDB and Spark Streaming

https://towardsdatascience.com/processing-time-series-data-in-real-time-with-influxdb-and-structured

Building a Sreaming application with spark, grafana, chronogram and influx:

https://medium.com/@xaviergeerinck/building-a-real-time-streaming-dashboard-with-spark-grafana-c

Performance Dashboard Spark and InfluxDB:

https://db-blog.web.cern.ch/blog/luca-canali/2019-02-performance-dashboard-apache-spark

Other alternatives for time series databases are: DalmatinerDB, InfluxDB, Prometheus, Riak TS, OpenTSDB, KairosDB

19.2.10 MPP Databases (Greenplum)

20 Data Processing and Analytics - Frameworks

20.1 Is ETL still relevant for Analytics?

Podcast Episode: #039 Is ETL Dead For Data Science & Big Data?	
Is ETL dead in Data Science and Big Data? In today's podcast I share with you my views on your questions regarding ETL (extract, transform, load). Is ETL still practiced or did pre-processing & cleansing replace it. What would replace ETL in Data Engineering	
YouTube	Click here to watch
Audio	Click here to listen

Table 20.1: Podcast: 039 Is ETL Dead for Data Science and Big Data?

20.2 Stream Processing

20.2.1 Three methods of streaming

In stream processing sometimes it is ok to drop messages, other times it is not. Sometimes it is fine to process a message multiple times, other times that needs to be avoided like hell.

Today's topic are the different methods of streaming: At most once, at least once and exactly once.

What this means and why it is so important to keep them in mind when creating a solution. That is what you will find out in this article.

20.2.2 At Least Once

At least once, means a message gets processed in the system once or multiple times. So with at least once it's not possible that a message gets into the system and is not getting processed.

It's not getting dropped or lost somewhere in the system.

One example where at least once processing can be used is when you think about a fleet management of cars. You get GPS data from cars and that data is transmitted with a timestamp and the GPS coordinates.

It's important that you get the GPS data at least once, so you know where the car is. If you're processing this data multiple times, it always has the the timestamp with it.

Because of that it does not matter that it gets processed multiple times, because of the timestamp. Or that it would be stored multiple times, because it would just override the existing one.

20.2.3 At Most Once

The second streaming method is at most once. At most once means that it's okay to drop some information, to drop some messages.

But it's important that a message is only only processed once as a maximum.

A example for this is event processing. Some event is happening and that event is not important enough, so it can be dropped. It doesn't have any consequences when it gets dropped.

But when that event happens it's important that it does not get processed multiple times. Then it would look as if the event happened five or six times instead of only one.

Think about engine misfires. If it happens once, no big deal. But if the system tells you it happens a lot you will think you have a problem with your engine.

20.2.4 Exactly Once

Another thing is exactly once, this means it's not okay to drop data, it's not okay to lose data and it's also not okay to process data multiple times.

An example for this is banking. When you think about credit card transactions it's not okay to drop a transaction.

When dropped your payment is not going through. It's also not okay to have a transaction processed multiple times, because then you are paying multiple times.

20.2.5 Check The Tools!

All of this sounds very simple and logical. What kind of processing is done has to be a requirement for your use case.

It needs to be thought about in the design process, because not every tool is supporting all three methods. Very often you need to code your application very differently based on the streaming method.

Especially exactly once is very hard to do.

So, the tool of data processing needs to be chosen based on if you need exactly once, at least once or if you need at most once.

20.3 MapReduce

Since the early days of the Hadoop eco system, the MapReduce framework is one of the main components of Hadoop alongside HDFS.

Google for instance used MapReduce to analyse stored HTML content of websites through counting all the HTML tags and all the words and combinations of them (for instance headlines). The output was then used to create the page ranking for Google Search.

That was when everybody started to optimise his website for the google search. Serious search engine optimisation was born. That was the year 2004.

How MapReduce is working is, that it processes data in two phases: The map phase and the reduce phase.

In the map phase, the framework is reading data from HDFS. Each dataset is called an input record.

Then there is the reduce phase. In the reduce phase, the actual computation is done and the results are stored. The storage target can either be a database or back HDFS or something else.

After all it's Java – so you can implement what you like.

The magic of MapReduce is how the map and reduce phase are implemented and how both phases are working together.

The map and reduce phases are parallelised. What that means is, that you have multiple map phases (mappers) and reduce phases (reducers) that can run in parallel on your cluster machines.

Here's an example how such a map and reduce process works with data:

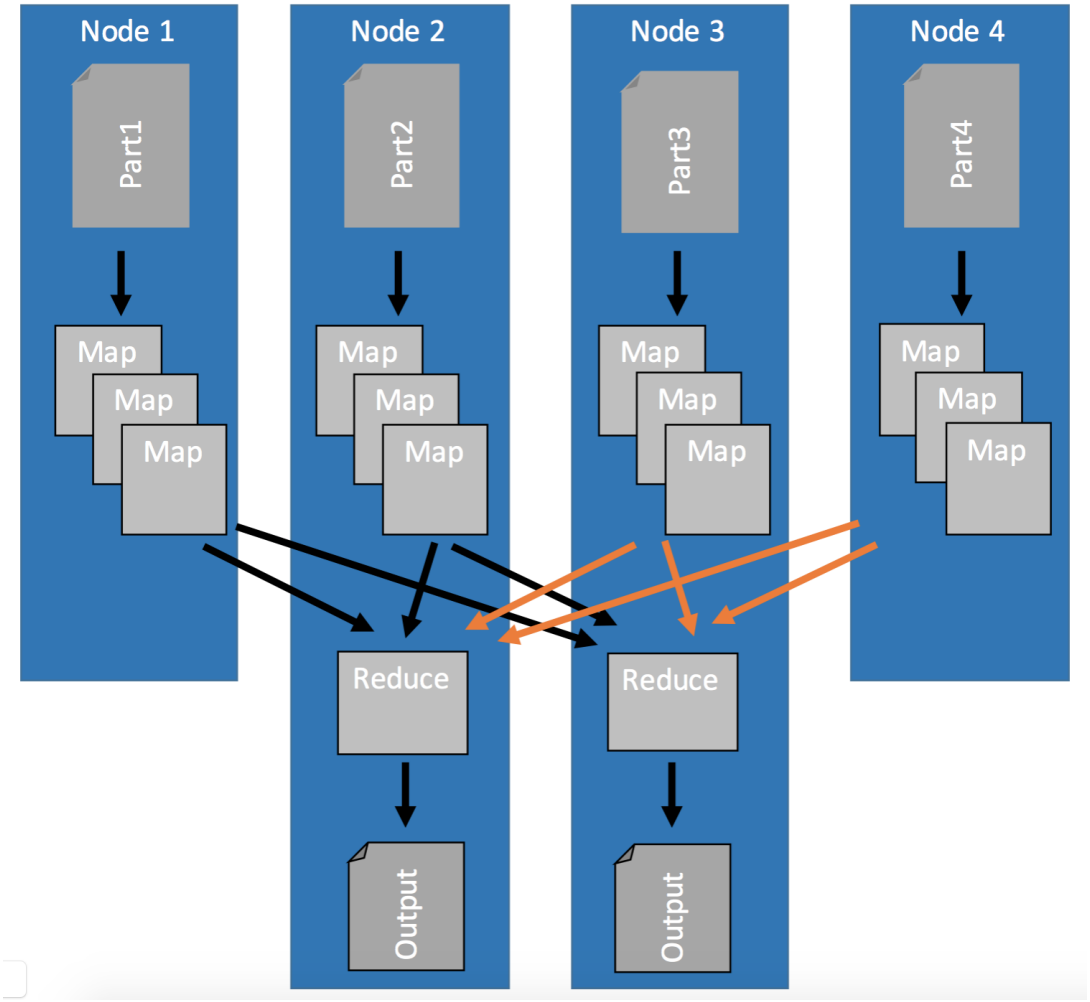

Figure 20.1: Mapping of input files and reducing of mapped records

20.3.1 How does MapReduce work

First of all, the whole map and reduce process relies heavily on using key-value pairs. That's what the mappers are for.

In the map phase input data, for instance a file, gets loaded and transformed into key-value pairs.

When each map phase is done it sends the created key-value pairs to the reducers where they are getting sorted by key. This means, that an input record for the reduce phase is a list of values from the mappers that all have the same key.

Then the reduce phase is doing the computation of that key and its values and outputting the results.

How many mappers and reducers can you use in parallel? The number of parallel map and reduce processes depends on how many CPU cores you have in your cluster. Every mapper and every reducer is using one core.

This means that the more CPU cores you actually have, the more mappers you can use, the faster the extraction process can be done. The more reducers you are using the faster the actual computation is being done.

To make this more clear, I have prepared an example:

20.3.2 Example

As I said before, MapReduce works in two stages, map and reduce. Often these stages are explained with a word count task.

Personally, I hate this example because counting stuff is to trivial and does not really show you what you can do with MapReduce. Therefore, we are going to use a more real world use-case from the IoT world.

IoT applications create an enormous amount of data that has to be processed. This data is generated by physical sensors who take measurements, like room temperature at 8 o'clock.

Every measurement consists of a key (the timestamp when the measurement has been taken) and a value (the actual value measured by the sensor).

Because you usually have more than one sensor on your machine, or connected to your system, the key has to be a compound key. Compound keys contain in addition to the measurement time information about the source of the signal.

But, let's forget about compound keys for now. Today we have only one sensor. Each measurement outputs key-value pairs like: Timestamp-Value.

The goal of this exercise is to create average daily values of that sensor's data.

The image below shows how the map and reduce process works.

First, the map stage loads unsorted data (input records) from the source (e.g. HDFS) by key and value (key:2016-05-01 01:02:03, value:1).

Then, because the goal is to get daily averages, the hour:minute:second information is cut from the timestamp.

That is all that happens in the map phase, nothing more.

After all parallel map phases are done, each key-value pair gets sent to the one reducer who is handling all the values for this particular key.

Every reducer input record then has a list of values and you can calculate (1+5+9)/3, (2+6+7)/3 and (3+4+8)/3. That's all.

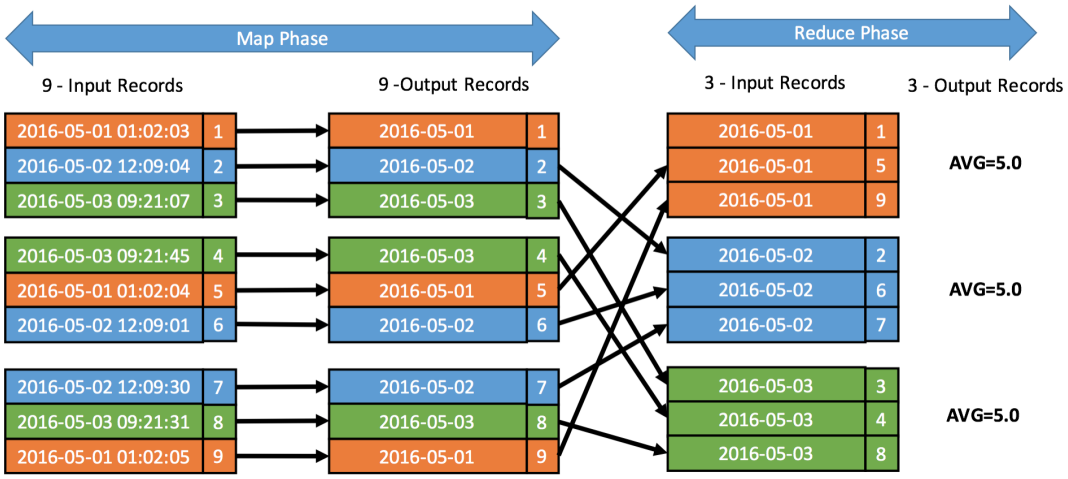

Figure 20.2: MapReduce Example of Time Series Data

What do you think you need to do to generate minute averages?

Yes, you need to cut the key differently. You then would need to cut it like this: "2016-05-01 01:02", keeping the hour and minute information in the key.

What you can also see is, why map reduce is so great for doing parallel work. In this case, the map stage could be done by nine mappers in parallel because each map is independent from all the others.

The reduce stage could still be done by three tasks in parallel. One for orange, one for blue and one for green.

That means, if your dataset would be 10 times as big and you'd have 10 times the machines, the time to do the calculation would be the same.

20.3.3 What is the limitation of MapReduce?

MapReduce is awesome for simpler analytics tasks, like counting stuff. It just has one flaw: It has only two stages Map and Reduce.

Figure 20.3: The Map Reduce Process

First MapReduce loads the data from HDFS into the mapping function. There you prepare the input data for the processing in the reducer. After the reduce is finished the results get written to the data store.

The problem with MapReduce is that there is no simple way to chain multiple map and reduce processes together. At the end of each reduce process the data must be stored somewhere.

This fact makes it very hard to do complicated analytics processes. You would need to chain MapReduce jobs together.

Chaining jobs with storing and loading intermediate results just makes no sense.

Another issue with MapReduce is that it is not capable of streaming analytics. Jobs take some time to spin up, do the analytics and shut down. Basically Minutes of wait time are totally normal.

This is a big negative point in a more and more real time data processing world.

20.4 Apache Spark

I talked about the three methods of data streaming in this podcast: https://anchor.fm/andreaskayy/embed/episodes/Three-Methods-of-Streaming-Data-e15r6o

20.4.1 What is the difference to MapReduce?

Spark is a complete in-memory framework. Data gets loaded from, for instance HDFS, into the memory of workers.

There is no longer a fixed map and reduce stage. Your code can be as complex as you want.

Once in memory, the input data and the intermediate results stay in memory (until the job finishes). They do not get written to a drive like with MapReduce.

This makes Spark the optimal choice for doing complex analytics. It allows you for instance to do iterative processes. Modifying a dataset multiple times in order to create an output is totally easy.

Streaming analytics capability is also what makes Spark so great. Spark has natively the option to schedule a job to run every X seconds or X milliseconds.

As a result, Spark can deliver you results from streaming data in “real time”.

20.4.2 How does Spark fit to Hadoop?

There are some very misleading articles out there titled ”Spark or Hadoop”, ”Spark is better than Hadoop” or even ”Spark is replacing Hadoop”.

So, it’s time to show you the differences between Spark and Hadoop. After this you will know when and for what you should use Spark and Hadoop.

You’ll also understand why ”Hadoop or Spark” is the totally wrong question.

20.4.3 Where’s the difference?

To make it clear how Hadoop differs from Spark I created this simple feature table:

	Storage	Analytics	Resource Management
Hadoop	Hadoop Distributed File System HDFS	MapReduce	YARN (Yet Another Resource Negotiator)
Spark	—-	Spark	Spark Resource Management

Figure 20.4: Hadoop vs Spark capabilities

Hadoop is used to store data in the Hadoop Distributed File System (HDFS). It can analyse the stored data with MapReduce and manage resources with YARN.

However, Hadoop is more than just storage, analytics and resource management. There’s a whole eco system of tools around the Hadoop core. I’ve written about tis eco system in this article: missing What is Hadoop and why is it so freakishly popular. You should check it out as well.

Compared to Hadoop, Spark is "just" an analytics framework. It has no storage capability. Although it has a standalone resource management, you usually don't use that feature.

20.4.4 Spark and Hadoop is a perfect fit

So, if Hadoop and Spark are not the same things, can they work together?

Absolutely! Here's how the first picture will look if you combine Hadoop with Spark:

missing

Figure 20.5: Combining Hadoop with Spark

As Storage you use HDFS. Analytics is done with Apache Spark and YARN is taking care of the resource management.

Why does that work so well together?

From a platform architecture perspective, Hadoop and Spark are usually managed on the same cluster. This means on each server where a HDFS data node is running, a Spark worker thread runs as well.

In distributed processing, network transfer between machines is a large bottle neck. Transferring data within a machine reduces this traffic significantly.

Spark is able to determine on which data node the needed data is stored. This allows a direct load of the data from the local storage into the memory of the machine.

This reduces network traffic a lot.

20.4.5 Spark on YARN:

You need to make sure that your physical resources are distributed perfectly between the services. This is especially the case when you run Spark workers with other Hadoop services on the same machine.

It just would not make sense to have two resource managers managing the same server's resources. Sooner or later they will get in each others way.

That's why the Spark standalone resource manager is seldom used.

So, the question is not Spark or Hadoop. The question has to be: Should you use Spark or MapReduce alongside Hadoop's HDFS and YARN.

20.4.6 My simple rule of thumb:

If you are doing simple batch jobs like counting values or doing calculating averages: Go with MapReduce.

If you need more complex analytics like machine learning or fast stream processing: Go with Apache Spark.

20.4.7 Available Languages

Spark jobs can be programmed in a variety of languages. That makes creating analytic processes very user-friendly for data scientists.

Spark supports Python, Scala and Java. With the help of SparkR you can even connect your R program to a Spark cluster.

If you are a data scientist who is very familiar with Python just use Python, its great. If you know how to code Java I suggest you start using Scala.

Spark jobs are easier to code in Scala than in Java. In Scala you can use anonymous functions to do processing.

This results in less overhead, it is a much cleaner, simpler code.

With Java 8 simplified function calls were introduced with lambda expressions. Still, a lot of people, including me prefer Scala over Java.

20.4.8 How Spark works: Driver, Executor, Sparkcontext

Podcast Episode: #100 Apache Spark Week Day 1	
On day one of the Apache Spark week we look into why Apache Spark is so great. We talk about stream and batch processing. What is the Spark Driver and Executors. Deployment modes Spark Standalone, YARN Mode and Kubernetes. We also talk a lot about Databricks in the about 30 minutes Q&A	
YouTube	Click here to watch

Table 20.2: Podcast: 100 Apache Spark Week Day 1

20.4.9 Spark batch vs stream processing

20.4.10 How does Spark use data from Hadoop

Another thing is data locality. I always make the point, that processing data locally where it is stored is the most efficient thing to do.

That's exactly what Spark is doing. You can and should run Spark workers directly on the data nodes of your Hadoop cluster.

Spark can then natively identify on what data node the needed data is stored. This enables Spark to use the worker running on the machine where the data is stored to load the data into the memory.

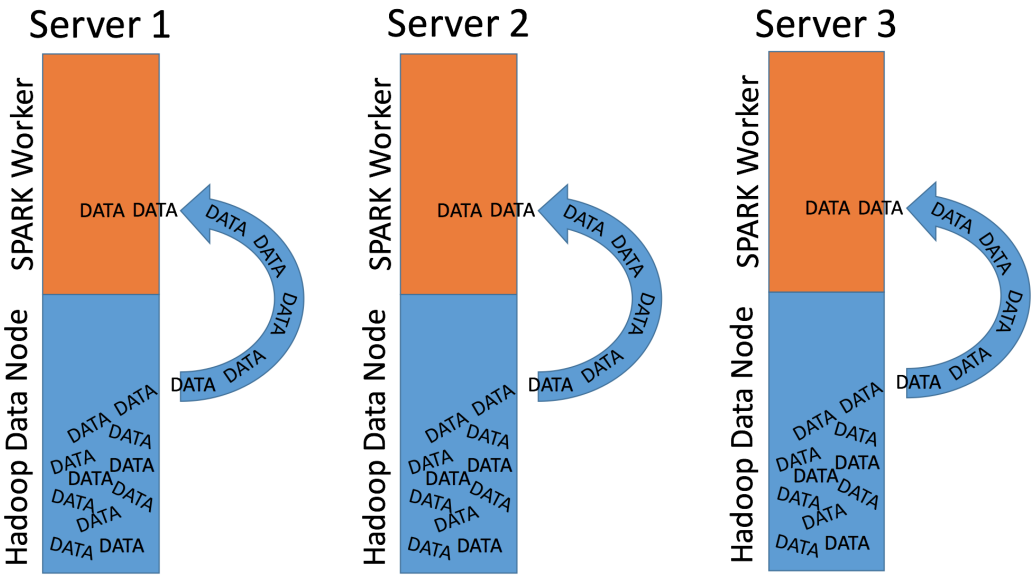

Figure 20.6: Spark Using Hadoop Data Locality

The downside of this setup is that you need more expensive servers. Because Spark processing needs stronger servers with more RAM and CPUs than a "pure" Hadoop setup.

20.4.11 What are RDDs and how to use them

RDDs are the core part of Spark. I learned and used RDDs first. It felt familiar coming from MapReduce. Nowadays you use Dataframes or Datasets.

I still find it valuable to learn how RDDs and therefore Spark works at a lower level.

Podcast Episode: #101 Apache Spark Week Day 2	
On day two of the Apache Spark week we take a look at major Apache Spark concepts: RDDs, transformations and actions, caching and broadcast variables.	
YouTube	Click here to watch

Table 20.3: Podcast: 101 Apache Spark Week Day 2

20.4.12 How and why to use SparkSQL?

When you use Apache Zeppelin notebooks to learn Spark you automatically come across SparkSQL. SparkSQL allows you to access Dataframes with SQL like queries.

Especially when you work with notebooks it is very handy to create charts from your data. You can learn from mistakes easier than just deploying Spark applications.

Podcast Episode: #102 Apache Spark Week Day 3	
We continue the Spark week, hands on. We do a full example from reading a csv, doing maps and flatmaps, to writing to disk. We also use SparkSQL to visualize the data.	
YouTube	Click here to watch

Table 20.4: Podcast: 102 Apache Spark Week Day 3

20.4.13 What are DataFrames how to use them

As I said before. Dataframes are the successors to RDDs. It's the new Spark API.

Dataframes are basically lake Tables in a SQL Database or like an Excel sheet. This makes them very simple to use and manipulate with SparkSQL. I highly recommend to go this route.

Processing with Dataframes is even faster then with RDDs, because it uses optimization alogrithms for the data processing.

Podcast Episode: #103 Apache Spark Week Day 4	
We look into Dataframes, Dataframes and Dataframes.	
YouTube	Click here to watch

Table 20.5: Podcast: 103 Apache Spark Week Day 4

20.4.14 Machine Learning on Spark? (Tensor Flow)

Wouldn't it be great to use your deep learning TensorFlow applications on Spark? Yes, it is already possible. Check out these Links:

Why do people integrate Spark with TensorFlow even if there is a distributed TensorFlow framework? https://www.quora.com/Why-do-people-integrate-Spark-with-TensorFlow-even-if-there-is

TensorFlow On Spark: Scalable TensorFlow Learning on Spark Clusters: https://databricks.com/session/tensorflow-on-spark-scalable-tensorflow-learning-on-spark-clusters

Deep Learning with Apache Spark and TensorFlow: https://databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html

20.4.15 MLlib:

The machine learning library MLlib is included in Spark so there is often no need to import another library.

I have to admit because I am not a data scientist I am not an expert in machine learning.

From what I have seen and read though the machine learning framework MLlib is a nice treat for data scientists wanting to train and apply models with Spark.

20.4.16 Spark Setup

From a solution architect's point of view Spark is a perfect fit for Hadoop big data platforms. This has a lot to do with cluster deployment and management.

Companies like Cloudera, MapR or Hortonworks include Spark into their Hadoop distributions. Because of that, Spark can be deployed and managed with the clusters Hadoop management web fronted.

This makes the process for deploying and configuring a Spark cluster very quick and admin friendly.

20.4.17 Spark Resource Management

When running a computing framework you need resources to do computation: CPU time, RAM, I/O and so on. Out of the box Spark can manage resources with it's stand-alone resource manager.

If Spark is running in an Hadoop environment you don't have to use Spark's own stand-alone resource manager. You can configure Spark to use Hadoop's YARN resource management.

Why would you do that? It allows YARN to efficiently allocate resources to your Hadoop and Spark processes.

Having a single resource manager instead of two independent ones makes it a lot easier to configure the resource management.

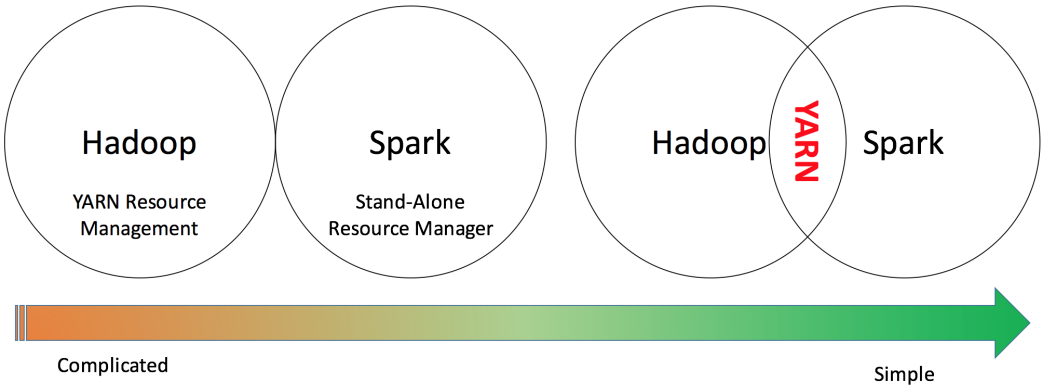

Figure 20.7: Spark Resource Management With YARN

20.5 Apache Nifi

Nifi is one of these tools that I identify as high potential tools. It allows you to create a data pipeline very easily.

Read data from a RestAPI and post it to Kafka? No problem Read data from Kafka and put it into a database? No problem

It's super versatile and you can do everything on the UI.

I use it in Part 3 of this Document. Check it out.

Check out the Apache Nifi FAQ website. Also look into the documentation to find all possible data sources and sinks of Nifi:

https://nifi.apache.org/faq.html

Here's a great blog about Nifi:

https://www.datainmotion.dev

20.6 StreamSets

https://youtu.be/djt8532UWow

https://www.youtube.com/watch?v=Qm5e574WoCU&t=2s

https://github.com/gschmutz/stream-processing-workshop/tree/master/04-twitter-data-ingestion-wit

https://streamsets.com/blog/streaming-data-twitter-analysis-spark/

21 Apache Kafka

21.1 Why a message queue tool?

21.2 Kafka architecture

21.3 What are topics

21.4 What does Zookeeper have to do with Kafka

21.5 How to produce and consume messages

My YouTube video how to set up Kafka at home: https://youtu.be/7F9tBwTUSeY

My YouTube video how to write to Kafka: https://youtu.be/RboQBZvZCh0

21.6 KAFKA Commands

Start Zookeeper container for Kafka:

```
docker run -d --name zookeeper-server    \
        --network app-tier    \
        -e ALLOW_ANONYMOUS_LOGIN=yes    \
        bitnami/zookeeper:latest
```

Start Kafka container:

```
docker run -d --name kafka-server   \
        --network app-tier   \
        -e KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper-server:2181   \
```

```
    -e ALLOW_PLAINTEXT_LISTENER=yes \
    bitnami/kafka:latest
```

22 Machine Learning

Podcast Episode: Machine Learning In Production	
Doing machine learning in production is very different than for proof of concepts or in education. One of the hardest parts is keeping models updated.	
Anchor	Click here to listen

Table 22.1: Podcast: Machine Learning In Production

22.1 How to do Machine Learning in production

Machine learning in production is using stream and batch processing. In the batch processing layer you are creating the models, because you have all the data available for training.

In the stream in processing layer you are using the created models, you are applying them to new data.

The idea that you need to incorporate is that it is a constant cycle. Training, applying, re-training, pushing into production and applying.

What you don't want to do is doing this manually. You need to figure out a process of automatic retraining and automatic pushing to into production of models.

In the retraining phase the system automatically evaluates the training. If the model no longer fits it works as long as it needs to create a good model.

After the evaluation of the model is complete and it's good, the model gets pushed into production. Into the stream processing.

22.2 Why machine learning in production is harder then you think

How to automate machine learning is something that drives me day in and day out.

What you do in development or education is, that you create a model and fit it to the data. Then that model is basically done forever.

Where I'm coming from, the IoT world, the problem is that machines are very different. They behave very different and experience wear.

22.3 Models Do Not Work Forever

Machines have certain processes that decrease the actual health of the machine. Machine wear is a huge issue. Models that that are built on top of a good machine don't work forever.

When the Machine wears out, the models need to be adjusted. They need to be maintained, retrained.

22.4 Where The Platforms That Support This?

Automatic re-training and re-deploying is a very big issue, a very big problem for a lot of companies. Because most existing platforms don't have this capability (I actually haven't seen one until now).

Look at AWS machine learning for instance. The process is: build, train, tune deploy. Where's the loop of retraining?

You can create models and then use them in production. But this loop is almost nowhere to be seen.

It is a very big issue that needs to be solved. If you want to do machine learning in production you can start with manual interaction of the training, but at some point you need to automate everything.

22.5 Training Parameter Management

To train a model you are manipulating input parameters of the models.

Take deep learning for instance.

To train you are manipulating for instance:

- How many layers do you use. - The depth of the layers, which means how many neurons you have in a layer. - What activation function you use, how long are you training and so on.

You also need to keep track of what data you used to train which model.

All those parameters need to be manipulated automatically, models trained and tested.

To do all that, you basically need a database that keeps track of those variables.

How to automate this, for me, is like the big secret. I am still working on figuring it out.

22.6 What's Your Solution?

Did you already have the problem of automatic re-training and deploying of models as well?

Were you able to use a cloud platform like Google, AWS or Azure?

It would be really awesome if you share your experience :)

22.7 How to convince people machine learning works

Many people still are not convinced that machine learning works reliably. But they want analytics insight and most of the time machine learning is the way to go.

This means, when you are working with customers you need to do a lot of convincing. Especially if they are not into machine learning themselves.

But it's actually quite easy.

22.8 No Rules, No Physical Models

Many people are still under the impression that analytics only works when it's based on physics. When there are strict mathematical rules to a problem.

Especially in engineering heavy countries like Germany this is the norm:

"Sere has to be a Rule for Everysing!" (imagine a German accent). When you're engineering you are calculating stuff based on physics and not based on data. If you are

constructing an airplane wing, you better make sure to use calculations so it doesn't fall off.

And that's totally fine.

Keep doing that!

Machine learning has been around for decades. It didn't quite work as good as people hoped. We have to admit that. But there is this preconception that it still doesn't work.

Which is not true: Machine learning works.

Somehow you need to convince people that it is a viable approach. That learning from data to make predictions is working perfectly.

22.9 You Have The Data. USE IT!

As a data scientist you have one ace up your sleeve, it's the obvious one:

It's the data and it's statistics.

You can use that data and those statistics to counter peoples preconceptions. It's very powerful if someone says: "This doesn't work"

You bring the data. You show the statistics and you show that it works reliably.

A lot of discussions end there.

Data doesn't lie. You can't fight data. The data is always right.

22.10 Data is Stronger Than Opinions

This is also why I believe that autonomous driving will come quicker than many of us think. Because a lot of people say, they are not safe. That you cannot rely on those cars.

The thing is: When you have the data you can do the statistics.

You can show people that autonomous driving really works reliably. You will see, the question of "Is this allowed or is this not allowed?" will be gone quicker than you think.

Because government agencies can start testing the algorithms based on predefined scenarios. They can run benchmarks and score the cars performance.

All those opinions, if it works, or if it doesn't work, they will be gone.

The motor agency has the statistics. The stats show people how good cars work.

Companies like Tesla, they have it very easy. Because the data is already there.

They just need to show us that the algorithms work. The end.

22.11 AWS Sagemaker

Train and apply models online with Sagemaker

Link to the OLX Slideshare with pros, cons and how to use Sagemaker: https://www.slideshare.net/mobile/AlexeyGrigorev/image-models-infrastructure-at-olx

23 Data Visualization

23.1 Android & IOS

23.2 How to design APIs for mobile apps

23.3 How to use Webservers to display content

This section does not contain any text that's why the page is messed up

23.3.1 Tomcat

23.3.2 Jetty

23.3.3 NodeRED

23.3.4 React

23.4 Business Intelligence Tools

23.4.1 Tableau

23.4.2 PowerBI

23.4.3 Quliksense

23.5 Identity & Device Management

23.5.1 What is a digital twin?

23.5.2 Active Directory

Part III

Data Engineering Course: Building A Data Platform

24 What We Want To Do

- Twitter data to predict best time to post using the hashtag datascience or ai
- Find top tweets for the day
- Top users
- Analyze sentiment and keywords

25 Thoughts On Choosing A Development Environment

For a local environment you need a good PC. I thought a bit about a budget build around 1.000 Dollars or Euros.

Podcast Episode: #068 How to Build a Budget Data Science PC	
In this podcast we look into configuring a sub 1000 dollar PC for data engineering and machine learning	
YouTube	Click here to watch
Audio	Click here to listen

Table 25.1: Podcast: 068 How to Build a Budget Data Science PC

26 A Look Into the Twitter API

Podcast Episode: #081 Twitter API Research	
In this podcast we were looking into how the Twitter API works and how you get access to it	
YouTube	Click here to watch

Table 26.1: Podcast: 081 Twitter API Research

27 Ingesting Tweets with Apache Nifi

Podcast Episode: #082 Reading Tweets With Apache Nifi & IaaS vs PaaS vs SaaS	
In this podcast we are trying to read Twitter Data with Nifi	
YouTube	Click here to watch

Table 27.1: Podcast: 082 Reading Tweets With Apache Nifi

Podcast Episode: #085 Trying to read Tweets with Nifi Part 2	
We are looking into the Big Data landscape chart and we are trying to read Twitter Data with Nifi again	
YouTube	Click here to watch

Table 27.2: Podcast: 085 Trying to read Tweets with Nifi Part 2

28 Writing from Nifi to Apache Kafka

Podcast Episode: #086 How to Write from Nifi to Kafka Part 1	
I've been working a lot on the cookbook, because it's so much fun. I gotta tell you what I added. Then we are trying to write the Tweets from Apache Nifi into Kafka. Also talk about Kafka basics.	
YouTube	Click here to watch

Table 28.1: Podcast: 086 How to Write from Nifi to Kafka Part 1

Podcast Episode: #088 How to Write from Nifi to Kafka Part 2	
In this podcast we finally figure out how to write to Kafka from Nifi. The problem was the network configuration of the Docker containers	
YouTube	Click here to watch

Table 28.2: Podcast: 088 How to Write from Nifi to Kafka Part 2

29 Apache Zeppelin

29.1 Install and Ingest Kafka Topic

Start the container:

```
docker run -d -p 8081:8080 --rm \
-v /Users/xxxx/Documents/DockerFiles/logs:/logs \
-v /Users/xxxx/Documents/DockerFiles/Notebooks:/notebook \
-e ZEPPELIN_LOG_DIR='/logs' \
-e ZEPPELIN_NOTEBOOK_DIR='/notebook' \
--network app-tier --name zeppelin apache/zeppelin:0.7.3
```

29.2 Processing Messages with Spark & SparkSQL

29.3 Visualizing Data

30 Switch Processing from Zeppelin to Spark

30.1 Install Spark

30.2 Ingest Messages from Kafka

30.3 Writing from Spark to Kafka

30.4 Move Zeppelin Code to Spark

Part IV

Case Studies

31 How I do Case Studies

31.1 Data Science @Airbnb

Podcast Episode: #063 Data Engineering At Airbnb Case Study	
How Airbnb is doing data engineering? Let's check it out.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.1: Podcast: 063 Data Engineering At Airbnb Case Study

Slides:

https://medium.com/airbnb-engineering/airbnb-engineering-infrastructure/home

Airbnb Engineering Blog: https://medium.com/airbnb-engineering

Data Infrastructure: https://medium.com/airbnb-engineering/data-infrastructure-at-airbnb-8adfb34f1

Scaling the serving tier: https://medium.com/airbnb-engineering/unlocking-horizontal-scalability-in-ou

Druid Analytics: https://medium.com/airbnb-engineering/druid-airbnb-data-platform-601c312f2a4c

Spark Streaming for logging events: https://medium.com/airbnb-engineering/scaling-spark-streaming-f

-Druid Wiki: https://en.wikipedia.org/wiki/Apache_Druid

31.2 Data Science @Amazon

https://www.datasciencecentral.com/profiles/blogs/20-data-science-systems-used-by-amazon-to-opera
https://aws.amazon.com/solutions/case-studies/amazon-migration-analytics/

31.3 Data Science @Baidu

https://www.slideshare.net/databricks/spark-sql-adaptive-execution-unleashes-the-power-of-cluster-in

31.4 Data Science @Blackrock

https://www.slideshare.net/DataStax/maintaining-consistency-across-data-centers-randy-fradin-black

31.5 Data Science @BMW

https://www.unibw.de/code/events-u/jt-2018-workshops/ws3_bigdata_vortrag_widmann.pdf

31.6 Data Science @Booking.com

Podcast Episode: #064 Data Engineering at Booking.com Case Study	
How Booking.com is doing data engineering? Let's check it out.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.2: Podcast: 064 Data Engineering At Booking.com Case Study

Slides:

https://www.slideshare.net/ConfluentInc/data-streaming-ecosystem-management-at-bookingcom?ref=https://www.confluent.io/kafka-summit-sf18/data-streaming-ecosystem-management

https://www.slideshare.net/SparkSummit/productionizing-behavioural-features-for-machine-learning-

https://www.slideshare.net/ConfluentInc/data-streaming-ecosystem-management-at-bookingcom?ref=https://www.confluent.io/kafka-summit-sf18/data-streaming-ecosystem-management

Druid: https://towardsdatascience.com/introduction-to-druid-4bf285b92b5a

Kafka Architecture: https://data-flair.training/blogs/kafka-architecture/

Confluent Platform: https://www.confluent.io/product/confluent-platform/

31.7 Data Science @CERN

Slides:

https://en.wikipedia.org/wiki/Large_Hadron_Collider

http://www.lhc-facts.ch/index.php?page=datenverarbeitung

Podcast Episode: #065 Data Engineering At CERN Case Study	
How is CERN doing Data Engineering? They must get huge amounts of data from the Large Hadron Collider. Let's check it out.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.3: Podcast: 065 Data Engineering At CERN Case Study

https://openlab.cern/sites/openlab.web.cern.ch/files/2018-09/2017_ESADE_Madrid_Big_Data.pdf

https://openlab.cern/sites/openlab.web.cern.ch/files/2018-05/kubeconeurope2018-cern-180507122303.pdf

https://www.slideshare.net/SparkSummit/next-cern-accelerator-logging-service-with-jakub-wozniak

https://databricks.com/session/the-architecture-of-the-next-cern-accelerator-logging-service

http://opendata.cern.ch

https://gobblin.apache.org

https://www.slideshare.net/databricks/cerns-next-generation-data-analysis-platform-with-apache-spar

https://www.slideshare.net/SparkSummit/realtime-detection-of-anomalies-in-the-database-infrastruct

31.8 Data Science @Disney

https://medium.com/disney-streaming/delivering-data-in-real-time-via-auto-scaling-kinesis-streams-7

31.9 Data Science @DLR

https://www.unibw.de/code/events-u/jt-2018-workshops/ws3_bigdata_vortrag_bamler.pdf

31.10 Data Science @Drivetribe

https://berlin-2017.flink-forward.org/kb_sessions/drivetribes-kappa-architecture-with-apache-flink/

https://www.slideshare.net/FlinkForward/flink-forward-berlin-2017-aris-kyriakos-koliopoulos-drivetrib

31.11 Data Science @Dropbox

https://blogs.dropbox.com/tech/2019/01/finding-kafkas-throughput-limit-in-dropbox-infrastructure/

31.12 Data Science @Ebay

https://www.slideshare.net/databricks/moving-ebays-data-warehouse-over-to-apache-spark-spark-as-
https://www.slideshare.net/databricks/analytical-dbms-to-apache-spark-auto-migration-framework-w

31.13 Data Science @Expedia

https://www.slideshare.net/BrandonOBrien/spark-streaming-kafka-best-practices-w-brandon-obrien
https://www.slideshare.net/Naveen1914/brandon-obrien-streamingdata

31.14 Data Science @Facebook

https://code.fb.com/core-data/apache-spark-scale-a-60-tb-production-use-case/

31.15 Data Science @Google

http://www.unofficialgoogledatascience.com/
https://ai.google/research/teams/ai-fundamentals-applications/
https://cloud.google.com/solutions/big-data/
https://datafloq.com/read/google-applies-big-data-infographic/385

31.16 Data Science @Grammarly

https://www.slideshare.net/databricks/building-a-versatile-analytics-pipeline-on-top-of-apache-spark-

31.17 Data Science @ING Fraud

https://sf-2017.flink-forward.org/kb_sessions/streaming-models-how-ing-adds-models-at-runtime-to-ca

31.18 Data Science @Instagram

https://www.slideshare.net/SparkSummit/lessons-learned-developing-and-managing-massive-300tb-ap

31.19 Data Science @LinkedIn

Podcast Episode: #073 Data Engineering At LinkedIn Case Study	
Let's check out how LinkedIn is processing data :)	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.4: Podcast: 073 Data Engineering At LinkedIn Case Study

Slides:

https://engineering.linkedin.com/teams/data#0

https://www.slideshare.net/yaelgarten/building-a-healthy-data-ecosystem-around-kafka-and-hadoop-l

https://engineering.linkedin.com/teams/data/projects/pinot

https://pinot.readthedocs.io/en/latest/intro.html#

https://towardsdatascience.com/building-machine-learning-at-linkedin-scale-f08bd9a63f0a

http://samza.apache.org

https://www.slideshare.net/ConfluentInc/more-data-more-problems-scaling-kafkamirroring-pipelines-a
ref=https://www.confluent.io/kafka-summit-sf18/more_data_more_problems

https://www.slideshare.net/KhaiTran17/conquering-the-lambda-architecture-in-linkedin-metrics-platf

https://www.slideshare.net/Hadoop_Summit/unified-batch-stream-processing-with-apache-samza

http://druid.io/docs/latest/design/index.html

31.20 Data Science @Lyft

https://eng.lyft.com/running-apache-airflow-at-lyft-6e53bb8fccff

Podcast Episode: #067 Data Engineering At NASA Case Study	
A look into how NASA is doing data engineering.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.5: Podcast: 067 Data Engineering At NASA Case Study

31.21 Data Science @NASA

Slides:

http://sites.nationalacademies.org/cs/groups/ssbsite/documents/webpage/ssb_182893.pdf

https://esip.figshare.com/articles/Apache_Science_Data_Analytics_Platform/5786421

http://www.socallinuxexpo.org/sites/default/files/presentations/OnSightCloudArchitecture-scale14x.pdf

https://www.slideshare.net/SparkSummit/spark-at-nasajplchris-mattmann?qid=90968554-288e-454a-v=&b=&from_search=4

https://en.m.wikipedia.org/wiki/Hierarchical_Data_Format

31.22 Data Science @Netflix

Podcast Episode: #062 Data Engineering At Netflix Case Study	
How Netflix is doing Data Engineering using their Keystone platform.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.6: Podcast: 062 Data Engineering At Netflix Case Study

Netflix revolutionized how we watch movies and TV. Currently over 75 million users watch 125 million hours of Netflix content every day!

Netflix's revenue comes from a monthly subscription service. So, the goal for Netflix is to keep you subscribed and to get new subscribers.

To achieve this, Netflix is licensing movies from studios as well as creating its own original movies and TV series.

But offering new content is not everything. What is also very important is, to keep you watching content that already exists.

To be able to recommend you content, Netflix is collecting data from users. And it is collecting a lot.

Currently, Netflix analyses about 500 billion user events per day. That results in a stunning 1.3 Petabytes every day.

All this data allows Netflix to build recommender systems for you. The recommenders are showing you content that you might like, based on your viewing habits, or what is currently trending.

The Netflix batch processing pipeline When Netflix started out, they had a very simple batch processing system architecture.

The key components were Chuckwa, a scalable data collection system, Amazon S3 and Elastic MapReduce.

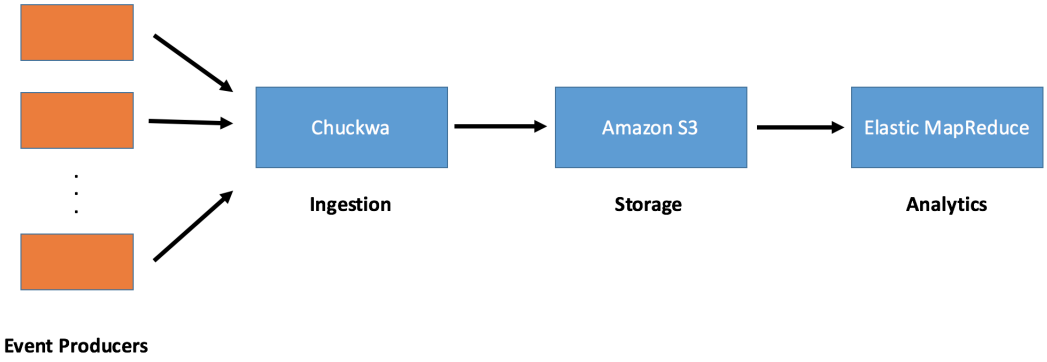

Figure 31.1: Old Netflix Batch Processing Pipeline

Chuckwa wrote incoming messages into Hadoop sequence files, stored in Amazon S3. These files then could be analysed by Elastic MapReduce jobs.

Netflix batch processing pipeline Jobs were executed regularly on a daily and hourly basis. As a result, Netflix could learn how people used the services every hour or once a day.

Know what customers want: Because you are looking at the big picture you can create new products. Netflix uses insight from big data to create new TV shows and movies.

They created House of Cards based on data. There is a very interesting TED talk about this you should watch:

How to use data to make a hit TV show — Sebastian Wernicke

Batch processing also helps Netflix to know the exact episode of a TV show that gets you hooked. Not only globally but for every country where Netflix is available.

Check out the article from TheVerge

They know exactly what show works in what country and what show does not.

It helps them create shows that work in everywhere or select the shows to license in different countries. Germany for instance does not have the full library that Americans have :(

We have to put up with only a small portion of TV shows and movies. If you have to select, why not select those that work best.

Batch processing is not enough As a data platform for generating insight the Cuckwa pipeline was a good start. It is very important to be able to create hourly and daily aggregated views for user behavior.

To this day Netflix is still doing a lot of batch processing jobs.

The only problem is: With batch processing you are basically looking into the past.

For Netflix, and data driven companies in general, looking into the past is not enough. They want a live view of what is happening.

The trending now feature One of the newer Netflix features is "Trending now". To the average user it looks like that "Trending Now" means currently most watched.

This is what I get displayed as trending while I am writing this on a Saturday morning at 8:00 in Germany. But it is so much more.

What is currently being watched is only a part of the data that is used to generate "Trending Now".

Figure 31.2: Netflix Trending Now Feature

"Trending now" is created based on two types of data sources: Play events and Impression events.

What messages those two types actually include is not really communicated by Netflix. I did some research on the Netflix Techblog and this is what I found out:

Play events include what title you have watched last, where you did stop watching, where you used the 30s rewind and others. Impression events are collected as you browse the Netflix Library like scroll up and down, scroll left or right, click on a movie and so on.

Basically, play events log what you do while you are watching. Impression events are capturing what you do on Netflix, while you are not watching something.

Netflix real-time streaming architecture Netflix uses three internet facing services to exchange data with the client's browser or mobile app. These services are simple Apache Tomcat based web services.

The service for receiving play events is called "Viewing History". Impression events are collected with the "Beacon" service.

The "Recommender Service" makes recommendations based on trend data available for clients.

Messages from the Beacon and Viewing History services are put into Apache Kafka. It acts as a buffer between the data services and the analytics.

Beacon and Viewing History publish messages to Kafka topics. The analytics subscribes to the topics and gets the messages automatically delivered in a first in first out fashion.

After the analytics the workflow is straight forward. The trending data is stored in a Cassandra Key-Value store. The recommender service has access to Cassandra and is making the data available to the Netflix client.

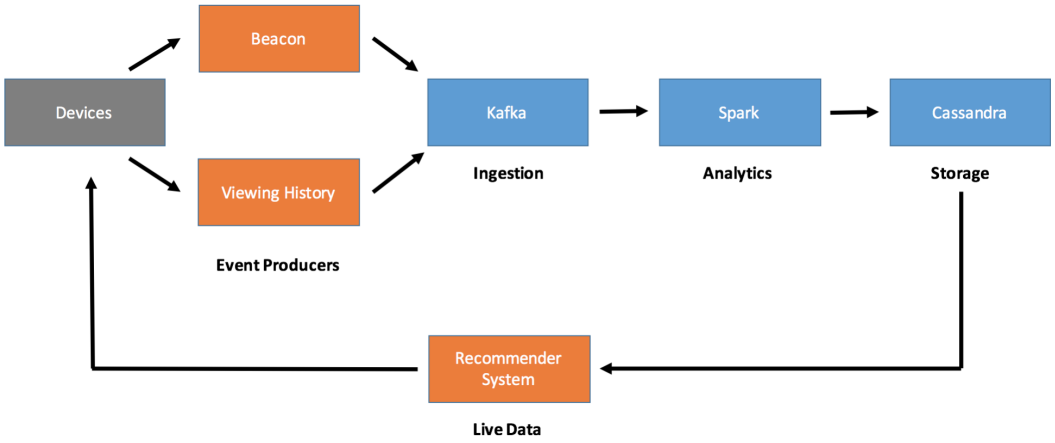

Figure 31.3: Netflix Streaming Pipeline

The algorithms how the analytics system is processing all this data is not known to the public. It is a trade secret of Netflix.

What is known, is the analytics tool they use. Back in Feb 2015 they wrote in the tech blog that they use a custom made tool.

They also stated, that Netflix is going to replace the custom made analytics tool with Apache Spark streaming in the future. My guess is, that they did the switch to Spark some time ago, because their post is more than a year old.

31.23 Data Science @OLX

Podcast Episode: #083 Data Engineering at OLX Case Study	
This podcast is a case study about OLX with Senior Data Scientist Alexey Grigorev as guest. It was super fun.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.7: Podcast: 083 Data Engineering at OLX Case Study

Slides:

https://www.slideshare.net/mobile/AlexeyGrigorev/image-models-infrastructure-at-olx

31.24 Data Science @OTTO

https://www.slideshare.net/SparkSummit/spark-summit-eu-talk-by-sebastian-schroeder-and-ralf-sigm

31.25 Data Science @Paypal

https://www.paypal-engineering.com/tag/data/

31.26 Data Science @Pinterest

Slides:

https://www.slideshare.net/ConfluentInc/pinterests-story-of-streaming-hundreds-of-terabytes-of-pins-
ref=https://www.confluent.io/kafka-summit-sf18/pinterests-story-of-streaming-hundreds-of-terabytes

Podcast Episode: #069 Engineering Culture At Pinterest	
In this podcast we look into data platform and processing at Pinterest.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.8: Podcast: 069 Engineering Culture At Pinterest

https://www.slideshare.net/ConfluentInc/building-pinterest-realtime-ads-platform-using-kafka-stream
ref=https://www.confluent.io/kafka-summit-sf18/building-pinterest-real-time-ads-platform-using-kafk

https://medium.com/@Pinterest_Engineering/building-a-real-time-user-action-counting-system-for-ad

https://medium.com/pinterest-engineering/goku-building-a-scalable-and-high-performant-time-series-

https://medium.com/pinterest-engineering/building-a-dynamic-and-responsive-pinterest-7d410e99f0a9

https://medium.com/@Pinterest_Engineering/building-pin-stats-25ec8460e924

https://medium.com/@Pinterest_Engineering/improving-hbase-backup-efficiency-at-pinterest-86159da

https://medium.com/@Pinterest_Engineering/pinterest-joins-the-cloud-native-computing-foundation-e

https://medium.com/@Pinterest_Engineering/using-kafka-streams-api-for-predictive-budgeting-9f58d2

https://medium.com/@Pinterest_Engineering/auto-scaling-pinterest-df1d2beb4d64

31.27 Data Science @Salesforce

https://engineering.salesforce.com/building-a-scalable-event-pipeline-with-heroku-and-salesforce-2549c

31.28 Data Science @Siemens Mindsphere

Podcast Episode: #059 What Is The Siemens Mindsphere IoT Platform?	
The Internet of things is a huge deal. There are many platforms available. But, which one is actually good? Join me on a 50 minute dive into the Siemens Mindsphere online documentation. I have to say I was super unimpressed by what I found. Many limitations, unclear architecture and no pricing available? Not good!	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.9: Podcast: 059 What Is The Siemens Mindsphere IoT Platform?

31.29 Data Science @Slack

https://speakerdeck.com/vananth22/streaming-data-pipelines-at-slack

31.30 Data Science @Spotify

Podcast Episode: #071 Data Engineering At Spotify Case Study	
In this episode we are looking at data engineering at Spotify, my favorite music streaming service. How do they process all that data?	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.10: Podcast: 071 Data Engineering At Spotify Case Study

Slides:

https://labs.spotify.com/2016/02/25/spotifys-event-delivery-the-road-to-the-cloud-part-i/

https://labs.spotify.com/2016/03/03/spotifys-event-delivery-the-road-to-the-cloud-part-ii/

https://labs.spotify.com/2016/03/10/spotifys-event-delivery-the-road-to-the-cloud-part-iii/

https://www.slideshare.net/InfoQ/scaling-the-data-infrastructure-spotify

https://www.datanami.com/2018/05/16/big-data-file-formats-demystified/

https://labs.spotify.com/2017/04/26/reliable-export-of-cloud-pubsub-streams-to-cloud-storage/

https://labs.spotify.com/2017/11/20/autoscaling-pub-sub-consumers/

31.31 Data Science @Symantec

https://www.slideshare.net/planetcassandra/symantec-cassandra-data-modelling-techniques-in-action

31.32 Data Science @Tinder

https://www.slideshare.net/databricks/scalable-monitoring-using-apache-spark-and-friends-with-utkar

Podcast Episode: #072 Data Engineering At Twitter Case Study	
How is Twitter doing data engineering? Oh man, they have a lot of cool things to share these tweets.	
YouTube	Click here to watch
Audio	Click here to listen

Table 31.11: Podcast: 072 Data Engineering At Twitter Case Study

31.33 Data Science @Twitter

Slides:

https://www.slideshare.net/sawjd/real-time-processing-using-twitter-heron-by-karthik-ramasamy

https://www.slideshare.net/sawjd/big-data-day-la-2016-big-data-track-twitter-heron-scale-karthik-ra

https://techjury.net/stats-about/twitter/

https://developer.twitter.com/en/docs/tweets/post-and-engage/overview

https://www.slideshare.net/prasadwagle/extracting-insights-from-data-at-twitter

https://blog.twitter.com/engineering/en_us/topics/insights/2018/twitters-kafka-adoption-story.html

https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twit
html

https://blog.twitter.com/engineering/en_us/topics/infrastructure/2019/the-start-of-a-journey-into-the
html

https://www.slideshare.net/billonahill/twitter-heron-in-practice

https://streaml.io/blog/intro-to-heron

https://www.youtube.com/watch?v=3QHGhnHx5HQ

https://hbase.apache.org

https://db-engines.com/en/system/Amazon+DynamoDB%3BCassandra%3BGoogle+Cloud+Bigtable%3BHBase

31.34 Data Science @Uber

https://eng.uber.com/uber-big-data-platform/

https://eng.uber.com/aresdb/

https://www.uber.com/us/en/uberai/

31.35 Data Science @Upwork

https://www.slideshare.net/databricks/how-to-rebuild-an-endtoend-ml-pipeline-with-databricks-and-u

31.36 Data Science @Woot

https://aws.amazon.com/de/blogs/big-data/our-data-lake-story-how-woot-com-built-a-serverless-data

31.37 Data Science @Zalando

Podcast Episode: #087 Data Engineering At Zalando Case Study Talk	
I had a great conversation about data engineering for online retailing with Michal Gancarski and Max Schultze. They showed Zalando's data platform and how they build data pipelines. Super interesting especially for AWS users.	
YouTube	Click here to watch

Table 31.12: Podcast: 087 Data Engineering At Zalando Case Study

Do me a favor and give these guys a follow on LinkedIn:

LinkedIn of Michal: https://www.linkedin.com/in/michalgancarski/

LinkedIn of Max: https://www.linkedin.com/in/max-schultze-b11996110/

Zalando has a tech blog with more infos and there is also a meetup in Berlin:

Zalando Blog: https://jobs.zalando.com/tech/blog/

Next Zalando Data Engineering Meetup: https://www.meetup.com/Zalando-Tech-Events-Berlin/events/262032282/

Interesting tools:

AWS CDK: https://docs.aws.amazon.com/cdk/latest/guide/what-is.html

Delta Lake: https://delta.io/

AWS Step Functions: https://aws.amazon.com/step-functions/AWSStateLanguage:https://states-language.net/spec.html

Youtube channel of the meetup: https://www.youtube.com/channel/UCxwul7aBm2LybbpKGbCOYNA playliststalkatSpark+AI

Summit about Zalando's Processing Platform: https://databricks.com/session/continuous-applications-

Talk at Strata London slides: https://databricks.com/session/continuous-applications-at-scale-of-100-t

https://jobs.zalando.com/tech/blog/what-is-hardcore-data-science--in-practice/?gh_src=4n3gxh1

https://jobs.zalando.com/tech/blog/complex-event-generation-for-business-process-monitoring-using-a

Part V

1001 Data Engineering Interview Questions

Looking for a job or just want to know what people find important? In this chapter you can find a lot of interview questions we collect on the stream.

Ultimately this should reach at least one thousand and one questions.

But Andreas, where are the answers?? Answers are for losers. I have been thinking a lot about this and the best way for you to prepare and learn is to look into these questions yourself.

This cookbook or Google will help you a long way. Some questions we discuss directly on the live stream.

32 Live Streams

First live stream where we started to collect these questions.

Podcast Episode: #096 1001 Data Engineering Interview Questions	
First live stream where we collect and try to answer as many interview questions as possible. If this helps people and is fun we do this regularly until we reach 1000 and one.	
YouTube	Click here to watch

Table 32.1: Podcast: 096 1001 Data Engineering Interview Questions

33 All Interview Questions

The interview questions are roughly structured like the sections in the "Basic data engineering skills" part. This makes it easier to navigate this document. I still need to sort them accordingly.

SQL DBs

- What are windowing functions?
- What is a stored procedure?
- Why would you use them?
- What are atomic attributes?
- Explain ACID props of a database
- How to optimize queries?
- What are the different types of JOIN (CROSS, INNER, OUTER)?
- What is the difference between Clustered Index and Non-Clustered Index - with examples?

The Cloud

- What is serverless?
- What is the difference between IaaS, PaaS and SaaS?
- How do you move from the ingest layer to the Cosumption layer? (In Serverless)
- What is edge computing?
- What is the difference between cloud and edge and on-premise?

Linux

- What is crontab?

Big Data

- What are the 4 V's?
- Which one is most important?

Kafka

- What is a topic?
- How to ensure FIFO?
- How do you know if all messages in a topic have been fully consumed?
- What are brokers?
- What are consumergroups?
- What is a producer?

Coding

- What is the difference between an object and a class?
- Explain immutability
- What are AWS Lambda functions and why would you use them?
- Difference between library, framework and package
- How to reverse a linked list
- Difference between args and kwargs
- Difference between OOP and functional programming

NoSQL DBs

- What is a key-value (rowstore) store?
- What is a columnstore?
- Diff between Row and col.store
- What is a document store?
- Difference between Redshift and Snowflake

Hadoop

- What file formats can you use in Hadoop?
- What is the difference between a name and a datanode?
- What is HDFS?
- What is the purpose of YARN?

Lambda Architecture

- What is streaming and batching?
- What is the upside of streaming vs batching?
- What is the difference between lambda and kappa architecture?
- Can you sync the batch and streaming layer and if yes how?

Python

- Difference between list tuples and dictionary

Data Warehouse & Data Lake

- What is a data lake?

- What is a data warehouse?
- Are there data lake warehouses?
- Two data lakes within single warehouse?
- What is a data mart?
- What is a slow changing dimension (types)?
- What is a surrogate key and why use them?

APIs (REST)

- What does REST mean?
- What is idempotency?
- What are common REST API frameworks (Jersey and Spring)?

Apache Spark

- What is an RDD?
- What is a dataframe?
- What is a dataset?
- How is a dataset typesafe?
- What is Parquet?
- What is Avro?
- Difference between Parquet and Avro
- Tumbling Windows vs. Sliding Windows
- Difference between batch and stream processing
- What are microbatches?

MapReduce

- What is a use case of mapreduce?
- Write a pseudo code for wordcount
- What is a combiner?

Docker & Kubernetes

- What is a container?
- Difference between Docker Container and a Virtual PC
- What is the easiest way to learn kubernetes fast?

Data Pipelines

- What is an example of a serverless pipeline?
- What is the difference between at most once vs at least once vs exactly once?
- What systems provide transactions?
- What is a ETL pipeline?

Airflow

- What is a DAG (in context of airflow/luigi)?
- What are hooks/is a hook?
- What are operators?
- How to branch?

DataVisualization

- What is a BI tool?

Security/Privacy

- What is Kerberos?
- What is a firewall?
- What is GDPR?
- What is anonymization?

Distributed Systems

- How clusters reach consensus (the answer was using consensus protocols like Paxos or Raft). Good I didnt have to explain paxos
- What is the cap theorem / explain it (What factors should be considered when choosing a DB?)
- How to choose right storage for different data consumers? It's always a tricky question

Apache Flink

- What is Flink used for?
- Flink vs Spark?

GitHub

- What are branches?
- What are commits?
- What's a pull request?

Dev/Ops

- What is continuous integration?

- What is continuous deployment?
- Difference CI/CD

Development / Agile

- What is Scrum?
- What is OKR?
- What is Jira and what is it used for?

Bibliography

[1] J. Ely and I. Stavrov1, *Analyzing chalk dust and writing speeds: computational and geometric approaches*, BoDine Journal of Mathematics **3** (2001), 14-159.

List of Figures

2.1 The Machine Learning Pipeline 13

12.1 Common SQL Platform Architecture 36
12.2 Scaling up a SQL Database 37
12.3 Scaling out a SQL Database 38

13.1 Platform Blueprint 40

14.1 Batch Processing Pipeline 44
14.2 Stream Processing Pipeline 45

16.1 Hadoop Ecosystem Components 49
16.2 Connections between tools 50
16.3 Flume Integration 51

19.1 HDFS Master and Data Nodes 61
19.2 Distribution of Blocks for a 512MB File 61

20.1 Mapping of input files and reducing of mapped records 69
20.2 MapReduce Example of Time Series Data 71
20.3 The Map Reduce Process 72
20.4 Hadoop vs Spark capabilities 73
20.5 Combining Hadoop with Spark 74
20.6 Spark Using Hadoop Data Locality 76
20.7 Spark Resource Management With YARN 79

31.1 Old Netflix Batch Processing Pipeline 105
31.2 Netflix Trending Now Feature 106
31.3 Netflix Streaming Pipeline 107

List of Tables

2.1 Podcast: 050 Data Engineer, Scientist or Analyst - Which One Is For You? 12
2.2 Podcast: 048 From Wannabe Data Scientist To Engineer My Journey . . 14

5.1 Podcast: 070 Engineering Culture At Spotify 22

10.1 Podcast: 082 Reading Tweets With Apache Nifi & IaaS vs PaaS vs SaaS 31
10.2 Podcast: 076 Cloud vs On-Premise 32

14.1 Podcast: 077 Lambda Architecture and Kappa Architecture 44
14.2 Podcast: 066 How To Do Data Science From A Data Engineers.. 46

15.1 Podcast: 055 Data Warehouse vs Data Lake 47

16.1 Podcast: 060 What Is Hadoop And Is Hadoop Still Relevant In 2019? . . 48

18.1 Podcast: 033 How APIs Rule The World 57
18.2 Podcast: 081 Twitter API Research 57

19.1 Podcast: 056 NoSQL Key Value Stores Explained with HBase 60
19.2 Podcast: 093 What is MongoDB 62
19.3 Podcast: What is Elasticsearch & Why is It So Popular? 63
19.4 Podcast: Druid NoSQL DB and Analytics DB Introduction 64

20.1 Podcast: 039 Is ETL Dead for Data Science and Big Data? 66
20.2 Podcast: 100 Apache Spark Week Day 1 75
20.3 Podcast: 101 Apache Spark Week Day 2 77
20.4 Podcast: 102 Apache Spark Week Day 3 77
20.5 Podcast: 103 Apache Spark Week Day 4 77

22.1 Podcast: Machine Learning In Production 83

25.1 Podcast: 068 How to Build a Budget Data Science PC 92

26.1 Podcast: 081 Twitter API Research 93

27.1 Podcast: 082 Reading Tweets With Apache Nifi 94

27.2 Podcast: 085 Trying to read Tweets with Nifi Part 2 94

28.1 Podcast: 086 How to Write from Nifi to Kafka Part 1 95
28.2 Podcast: 088 How to Write from Nifi to Kafka Part 2 95

31.1 Podcast: 063 Data Engineering At Airbnb Case Study 99
31.2 Podcast: 064 Data Engineering At Booking.com Case Study 100
31.3 Podcast: 065 Data Engineering At CERN Case Study 101
31.4 Podcast: 073 Data Engineering At LinkedIn Case Study 103
31.5 Podcast: 067 Data Engineering At NASA Case Study 104
31.6 Podcast: 062 Data Engineering At Netflix Case Study 104
31.7 Podcast: 083 Data Engineering at OLX Case Study 108
31.8 Podcast: 069 Engineering Culture At Pinterest 109
31.9 Podcast: 059 What Is The Siemens Mindsphere IoT Platform? 109
31.10 Podcast: 071 Data Engineering At Spotify Case Study 110
31.11 Podcast: 072 Data Engineering At Twitter Case Study 111
31.12 Podcast: 087 Data Engineering At Zalando Case Study 112

32.1 Podcast: 096 1001 Data Engineering Interview Questions 116