Institute of Data

2023

Data Science and AI

Module 10

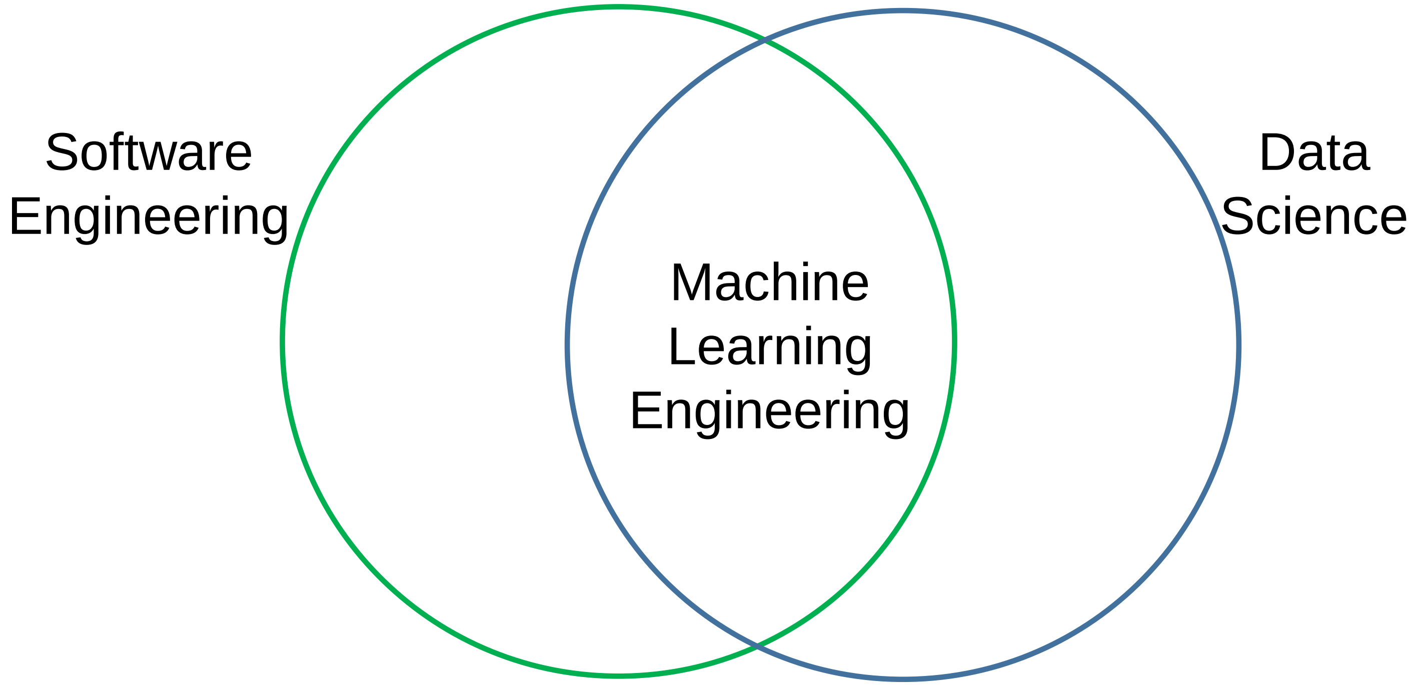
Machine Learning Deployment and Cloud Computing

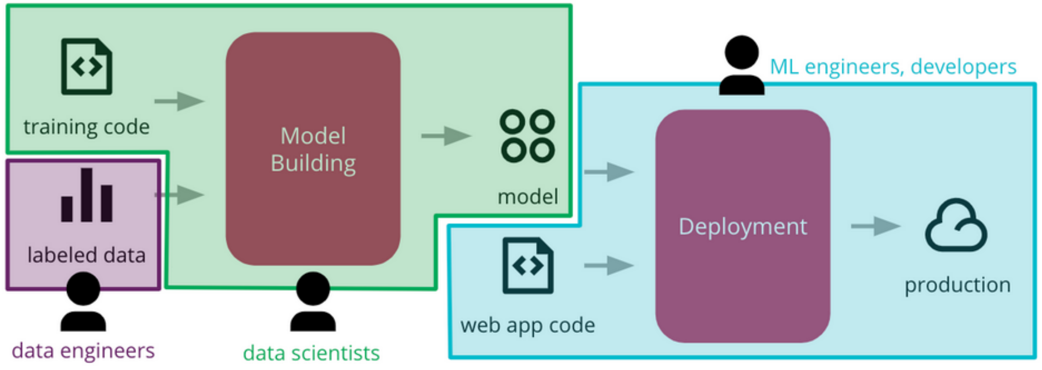
Agenda: Module 10 – Machine Learning Deployment and Cloud Computing

- Introduction
- Cloud computing
- Deployment
- DevOps, AutoML
- Cluster Computing
- Streaming Data

What is Machine Learning Deployment?

The process of making machine learning models and analyses easier to use and more accessible, bridging the gap between models and consumers





https://martinfowler.com/articles/cd4ml.html

Key Machine Learning Engineering Skills

- Well-versed in Cloud Computing and Big Data technologies such as Hadoop or Spark
- Machine learning – knowledge of algorithms and how to code them using libraries (e.g. sklearn)
- Data engineering – automating the processes of feeding data in and feature engineering/selection
- Software engineering
    - scaling out models into a "production environment" where it is in operation for consumption by their intended users
    - optimising code for performance

Cloud Computing

- Basics
- Virtualisation
- Cloud Computing Service Models
  - SaaS, PaaS, IaaS
- Introduction to AWS
  - S3
  - Machine Learning Services

The Challenges of Real-World Data and Computation

- "Big" Data is more than volume – it is about its complexity making it untenable to process using traditional means
  - **Volume** (requires multiple machines to store and process)
  - **Velocity** (data in motion)
  - **Variety** (often unstructured)
  - **Veracity** (often unreliable or incomplete)
- Challenge for data science – making models work in a dynamic real-time environment
  - Changes to data, algorithms, code
- Utilising a **cloud computing** environment addresses some of these challenges

Cloud Computing Basics

- **Cloud Computing** – the delivery of IT services (e.g. software, storage, computing processors) over a computer network (usually the Internet or Intranet)
- Characteristics (as per NIST – National Institute of Standards and Technology)
  - on-demand self-service provisioning of resources
  - broad network access
  - resource pooling (grouping together resources to improve the quality of service)
  - rapid elasticity (ability to scale up or down easily)
  - measured service (usage of resources is calculated and billed accordingly)
- Typically makes use of **virtualisation**

Virtualisation

- **Virtualisation** allows multiple operating systems to be run on the same physical hardware of a computer
- A **virtual machine** (VM) is a software program or operating system running on hardware that behaves as though it were a single computer that runs its own applications
  - Multiple virtual machine **instances** may exist on the same physical host machine
  - If an application crashes on one VM it does not affect other applications on different VMs
- Allows for a more efficient use of computing resources

Cloud Computing Service Models

- **SaaS** – entire application made available over a web browser (e.g. Gmail, Office 365)
- **PaaS** – software development and deployment (e.g. AWS Elastic Beanstalk, Heroku, Render, Google App Engine)
- **IaaS** – provides storage, networking, virtualised hardware (e.g. AWS, Microsoft Azure, Google Compute Engine)

On-site	IaaS	PaaS	SaaS
Applications (You manage)	Applications (You manage)	Applications (You manage)	Applications (Service provider manages)
Data (You manage)	Data (You manage)	Data (You manage)	Data (Service provider manages)
Runtime (You manage)	Runtime (You manage)	Runtime (Service provider manages)	Runtime (Service provider manages)
Middleware (You manage)	Middleware (You manage)	Middleware (Service provider manages)	Middleware (Service provider manages)
O/S (You manage)	O/S (You manage)	O/S (Service provider manages)	O/S (Service provider manages)
Virtualization (You manage)	Virtualization (Service provider manages)	Virtualization (Service provider manages)	Virtualization (Service provider manages)
Servers (You manage)	Servers (Service provider manages)	Servers (Service provider manages)	Servers (Service provider manages)
Storage (You manage)	Storage (Service provider manages)	Storage (Service provider manages)	Storage (Service provider manages)
Networking (You manage)	Networking (Service provider manages)	Networking (Service provider manages)	Networking (Service provider manages)

You manage

Service provider manages

https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas

Instance Specifications

- Number and Type of processor (CPU vs GPU vs TPU)
- Memory (RAM or Storage)
- Type of Operating System (Linux/Unix, MacOS, Windows)
- Bandwidth
- Instances optimised for Compute, Memory (large datasets in memory) or Storage

- AWS Example: https://aws.amazon.com/ec2/instance-types/

Leading Cloud Computing providers

	File storage	Computation	List of services
aws	S3	EC2	https://aws.amazon.com/products/
Azure	File, Queue, Table, Blob (Storage Types)	Azure VM	https://azure.microsoft.com/en-au/services/
	Google Cloud Storage	Google Compute Engine	https://cloud.google.com/products

AWS S3 Buckets

- Simple Storage Service – does not provide computation
- **Buckets** store objects (data) and associated metadata
- Files are referenced as keys – there is no hierarchical structure
  - e.g. though dir/f1 and dir/f2 are allowed filenames, they do not share a common folder dir
- Its name should be globally unique
- Pricing: https://aws.amazon.com/s3/pricing/

Amazon Web Services

Machine Learning

Build with powerful services and platforms, and the broadest machine learning framework support anywhere.

Learn More

Analytics & Data Lakes

Securely store, categorize, and analyze all your data in one, centralized repository.

Learn More

Internet of Things

A system of ubiquitous devices connecting the physical world to the cloud.

Learn More

Serverless Computing

Build and run applications and services without thinking about servers.

Learn More

Containers

Package and deploy applications that are lightweight and provide a consistent, portable software environment for applications to easily run and scale anywhere.

Learn More

Enterprise Applications

Build with a mature set of services specifically designed for the unique security, compliance, privacy, and governance requirements of large organizations.

Learn More

Storage

Durable, cost-effective options for backup, disaster recovery, and data archiving at petabyte scale.

Learn More

Windows Workloads

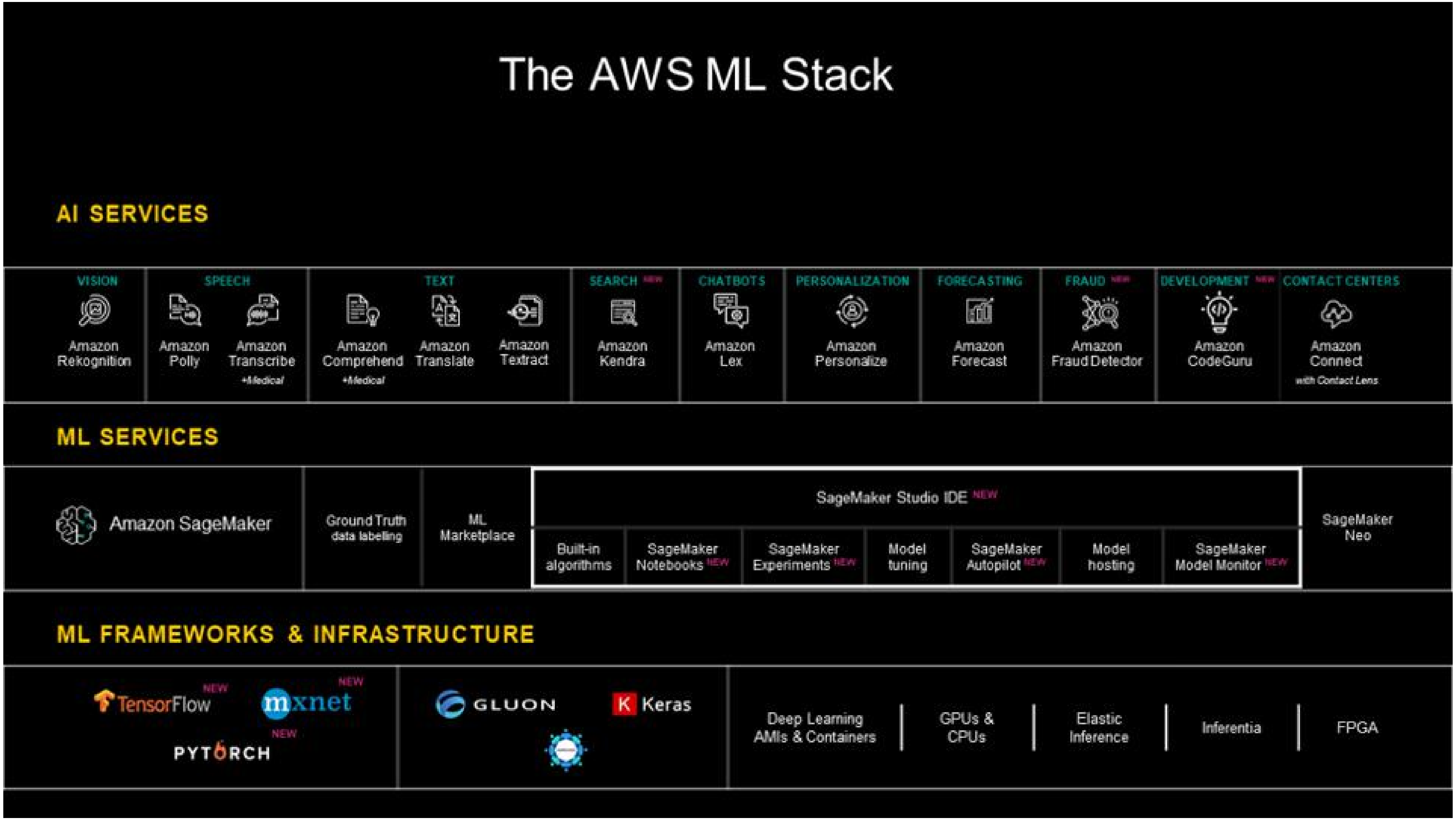
Flexible, scalable compute capacity for Microsoft applications. Easily manage and secure Windows workloads.

Learn More

https://aws.amazon.com/

AWS for Machine Learning

- **SageMaker** – notebooks (Jupyter and JupyterLab)
- **Amazon Rekognition** – image and video analysis such as facial recognition
- **Amazon Comprehend** – NLP
- **Amazon Textract** – text extraction from images or pdf files
- Amazon Lex – Building conversational interfaces (e.g. chatbots) into an application using voice and text
- Amazon Polly – text to speech
- Amazon Transcribe – speech to text
- AWS DeepLens – Deep-Learning-enabled video camera
- Amazon Personalize – individual recommendations
- Amazon Forecast – time series forecasting

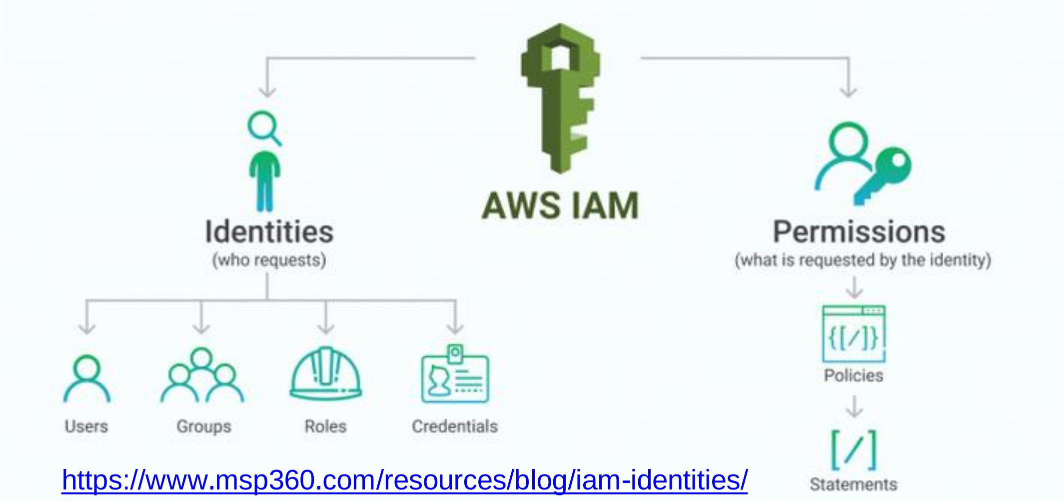


https://www.forbes.com/sites/moorinsights/2020/03/04/surprise-aws-leads-in-cloud-ai-services-ranking/

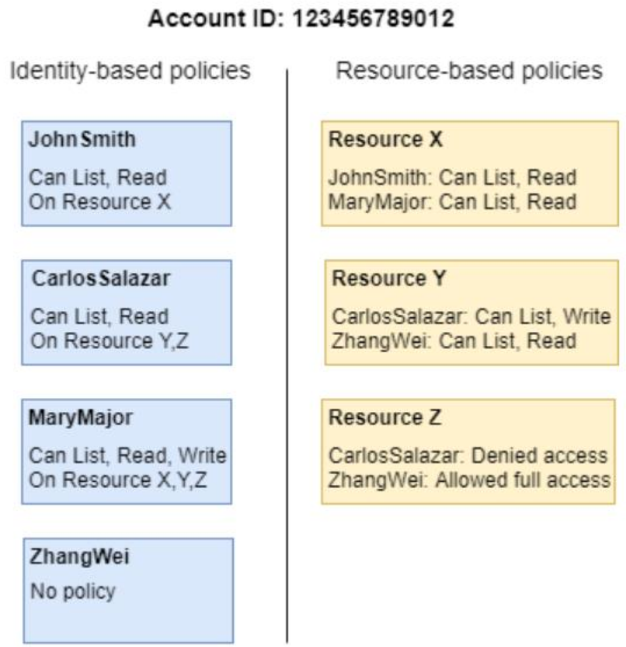
Identity and Access Management (IAM)

- Access in AWS is managed by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources.

- **User** – an individual
- **Group** – a set of users sharing common roles and policies
- **Role** – a set of permissions for granting access to resources – not associated with any specific user or group and less permanent
- **Policy** – a document listing permissions (e.g. creating or updating a database table)



https://www.msp360.com/resources/blog/iam-identities/



https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html

Lab 10.1: AWS SageMaker and Managed Services

Purpose

- Gain familiarity with AWS SageMaker and some Machine Learning Managed Services via the Console

Resources

- COCO Dataset (images)

Materials

- Jupyter Notebook (Lab-10_1)
- Sample image files

Deployment

- Challenges
- Considerations
- Pipelining
- Small-scale deployment via a REST API
- DevOps and CI/CD
- Monitoring

Deployment

- **Deployment** refers to making models available in production environments, where they can provide exposure to more users or be consumed by other software systems.
- Only after this deployment do models add business value.
- Types of deployment depend on the frequency of model training:
- **Batch** – large amount of data is required for a prediction (e.g. a user’s entire transaction history), latency does not matter
- **Real-time** – a small amount of data is required (e.g. choosing an online ad to show to a customer)

Challenges of Deployment

- Requires coordination between data scientists, IT teams, software developers, and business professionals to ensure a model works reliably in the organisation's production environment
- Sometimes a model is written in one language (e.g. R, Python) but is deployed in another (e.g. Java, C++)
- Incompatibility of code across different machines (operating systems, versions of libraries etc)
  - -> Containerisation

Challenges of Deployment

- Processing power (e.g. GPUs, TPUs) might be scarce and expensive for deployment at a large scale
- Scaling to accommodate a large number of users
- Data leakage or contamination – information about a target variable is present when it should not (e.g. information from the future, features depend on data from the holdout set)
- Reproducibility of results

Deployment Considerations

- Will the deployment deliver business value? Success metrics?
- What software tools to use?
- What are the dependencies (for code and data)?
- How will the results be consumed?
  - A dashboard?
  - Linked to an alert or triggering another software process?

Deployment Considerations

- Hosted on the cloud or on-premise?
- How to scale it to more users?
- How to guarantee security?
- How often should the model be retrained?
- How to monitor performance?

Model pipelining

- To ensure reproducibility avoid transforming data manually (e.g. find/replace operations, editing a Spreadsheet in Excel)

- Make use of pipelines consisting of steps such as the following
  - Read in the data (possibly using a feature store)
  - Merge multiple sources
  - Completing missing values
  - Address class imbalance (e.g. resampling)
  - Feature selection/engineering
  - Dimension Reduction
  - Model training with optimised hyperparameters

- In Python: from sklearn.pipeline import Pipeline

Small-Scale Deployment

- Easiest way is to create a web REST API endpoint using a Web framework such as Flask
  - Enables real-time predictions but would not scale up to a large number of users

Basic steps:

- Set up a reproducible pipeline from data to prediction
- Save the model pipeline in binary format (pickle or joblib)
- Build an image (container) with model and library dependencies
- Have the image hosted on a Platform as a Service (e.g. Heroku, Render, AWS Elastic Beanstalk)

Model as Data – what needs to be stored?

- Persistence – saving a model for future use without having to retrain

- Linear regression – a list of coefficients
  - Predict by applying a linear combination of the inputs
- KNN – the entire dataset
  - Predict by finding the k nearest neighbours to a given input
- Decision tree – the set of decisions at each node
  - Predict by traversing the tree according to the outcome of each decision
- Neural network – all weights and activation functions between neurons
  - Predict by applying the linear combinations and activation functions to a given input

Containers

- A container is a set of software processes isolated from the rest of an operating system

- It contains all the files necessary to support the processes (model + library dependencies)

- Different from a virtual machine in that they do not require a full operating system – a single operating system may run separate container instances

- Programs inside a container only have visibility to contents of the container and devices connected to it

Containers

- Advantage: avoiding environment-related issues - if containerised code works on one machine, it will run on another irrespective of the characteristics of the machine

- Docker is the most popular containerization service

- Docker Tutorial: docker-curriculum.com

Deployment via Flask

- Flask is a web framework allowing one to create small-scale web apps in Python
- Allows one to create a REST API service that is deployed on a web server

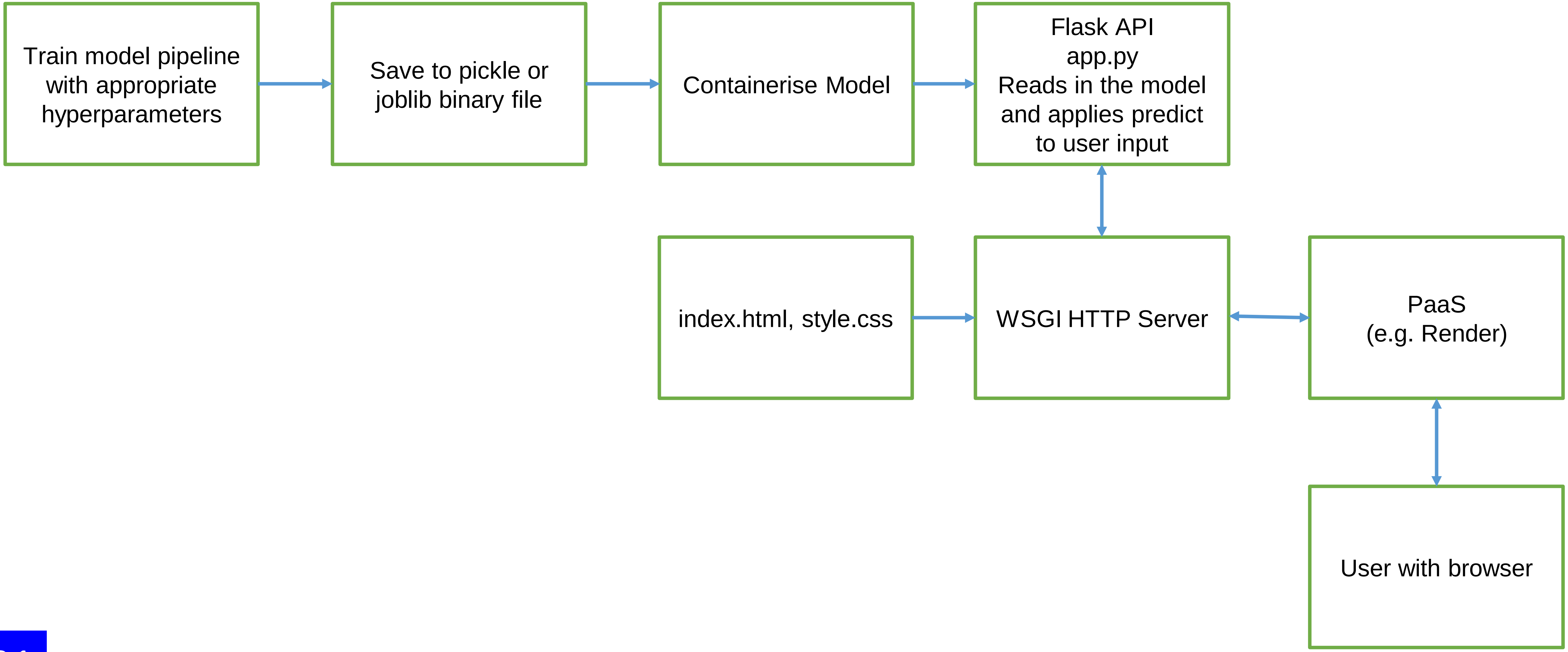
- For larger applications one can use Gunicorn with Nginx as the web server

```
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    …
```

@app.route allows Flask to assign functions to URLs

Deployment via Flask



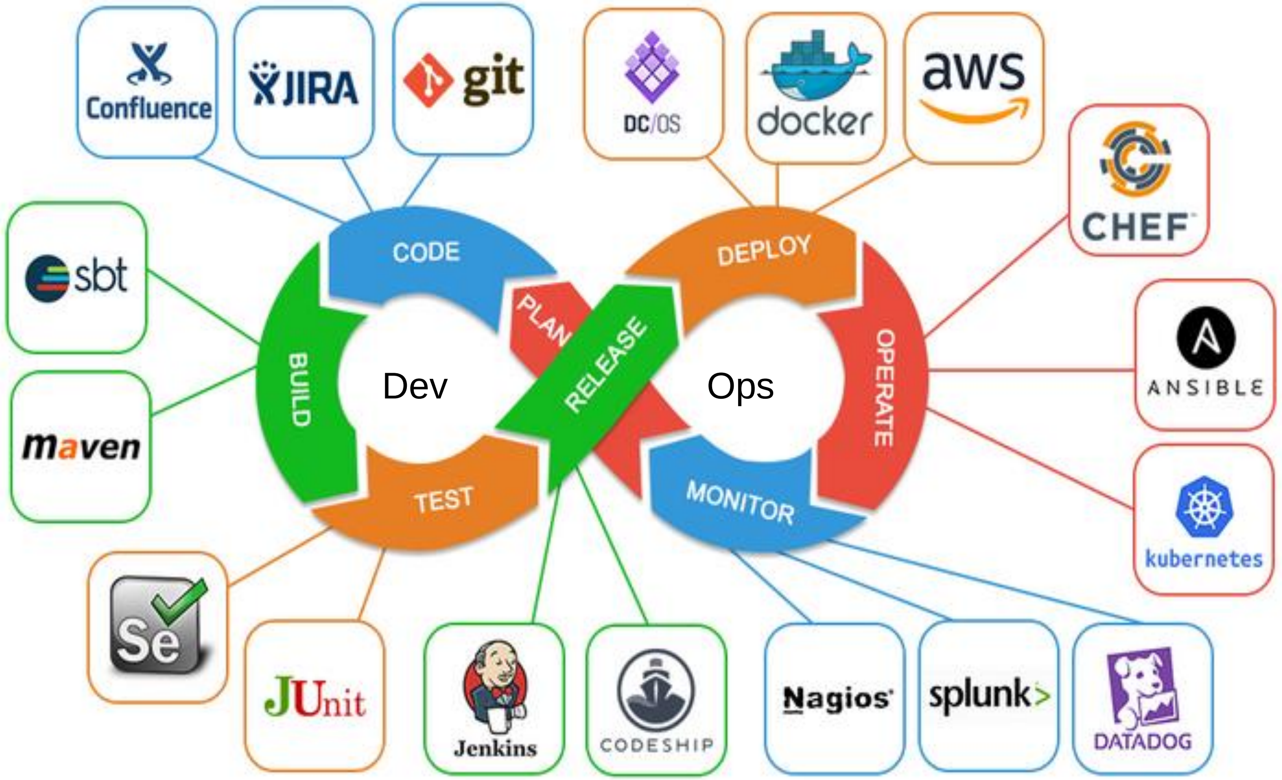
Model Monitoring and Maintenance

- Monitoring and scoring models while in production ensures they are behaving as intended
- Model/data drift – the statistics of input data no longer match the training data – hence the need to retrain
  - Statistical tests can check for the closeness between distributions of training data and data observed in production
- Concept drift – change in the meaning/interpretation of the target variable over time
  - e.g. in detecting rare events the definition of an anomaly changes over time
- Usually the remedy is more frequent training but on occasions the model needs to be redesigned

Larger scale: DevOps and CI/CD

- DevOps is a combination of Software Development (Dev) and IT Operations (Ops)
- IT Operations – services provided by an IT department for its customers
- DevOps aims to shorten the systems development life cycle and provide continuous delivery while maintaining high software quality
- CI/CD (Continuous Integration/Continuous Deployment) automates the process of building-testing-deploying (e.g in Dev/Test/Prod environments), shortening the life cycle of software development while maintaining quality
- New ideas can be deployed in production more rapidly, providing value to the user

The DevOps Lifecycle with Sample Tools

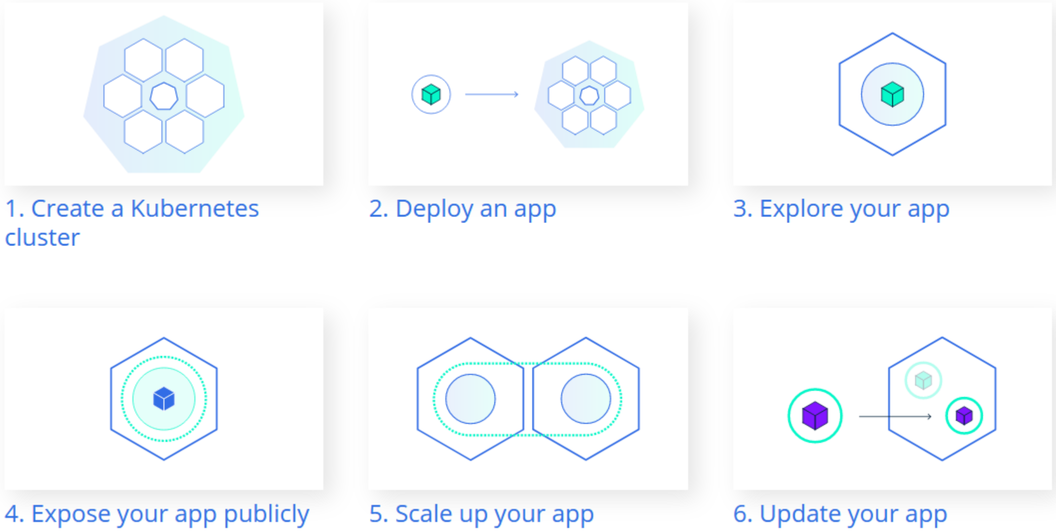


https://dzone.com/articles/why-should-i-learn-devops

See also: https://i.redd.it/92ja1d3v0x141.jpg

Orchestration

- A web service can scale by orchestration – a cluster of containers can be created on demand based on a copy of the original image container
- This also ensures fault tolerance should one container fail
- Kubernetes and Amazon ECS (Elastic Container Service) are well known orchestration systems



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Enabling Security

Ways to ensure secure model deployment include

- Cryptographically encrypting the model
- Hosting with a secure protocol (https vs http)
- Providing access control (e.g. token-based API)
- Including password protections

Machine Learning Platforms

Developed internally

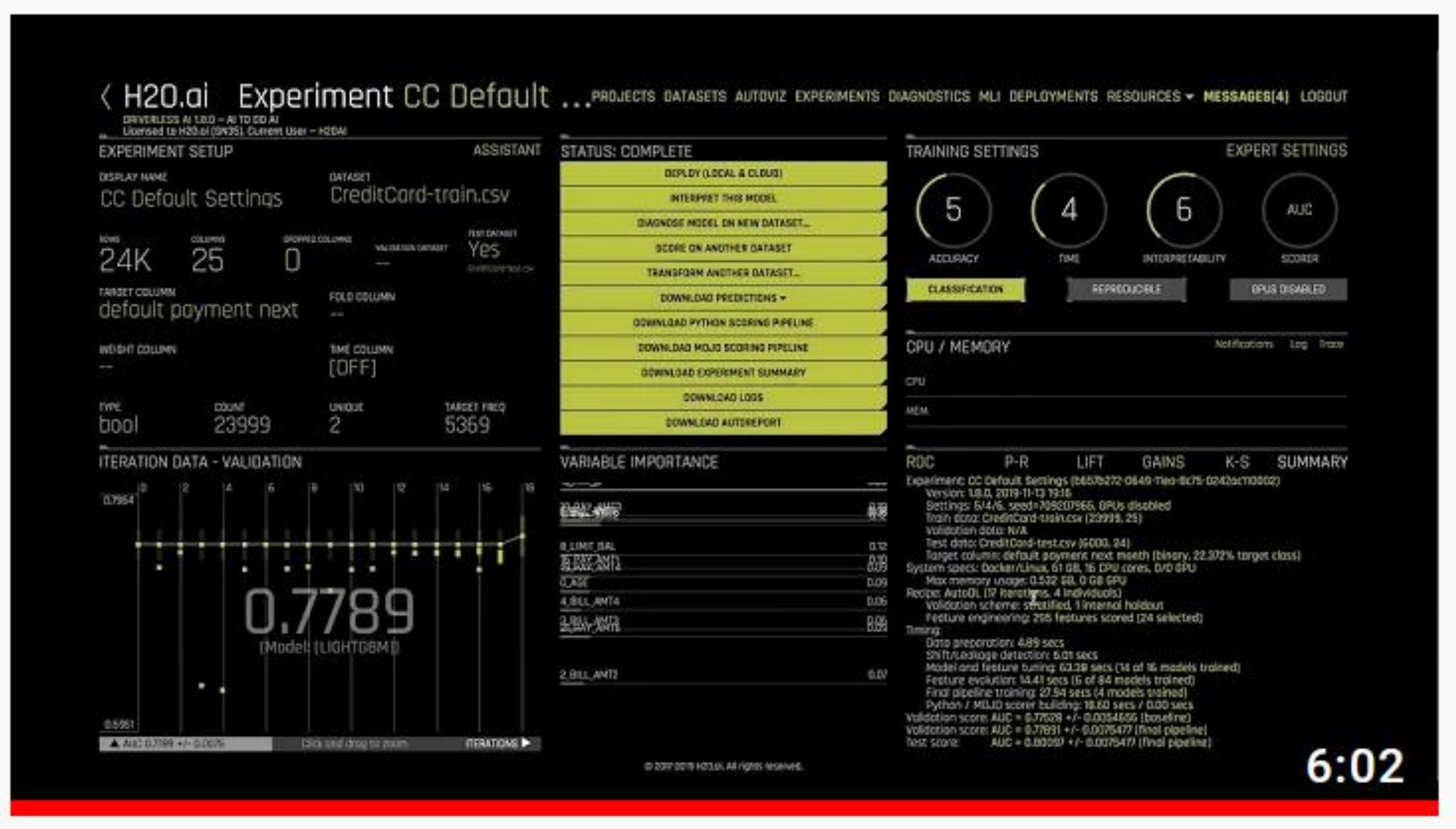
- Google: TFX (TensorFlow Extended)
- Uber: Michelangelo
- Databricks: MLFlow
- Facebook: FBLearner Flow

AutoML

- Tools exist to automate the process from data ingestion to deployment
  - Most useful for data clean-up and hyperparameter tuning
- Data scientists are still needed for problem definition, appropriate feature engineering requiring domain knowledge

- Google Cloud AutoML
- Microsoft Azure AutoML
- AWS SageMaker Autopilot
- DataRobot
- H20.ai
- TPOT
- Auto-Sklearn

One example of AutoML - H2O.ai demo



https://www.youtube.com/watch?v=ZqCoFp3-rGc

Lab 10.2: Model Deployment

- Purpose
  - Learn how to deploy a simple app on a cloud platform

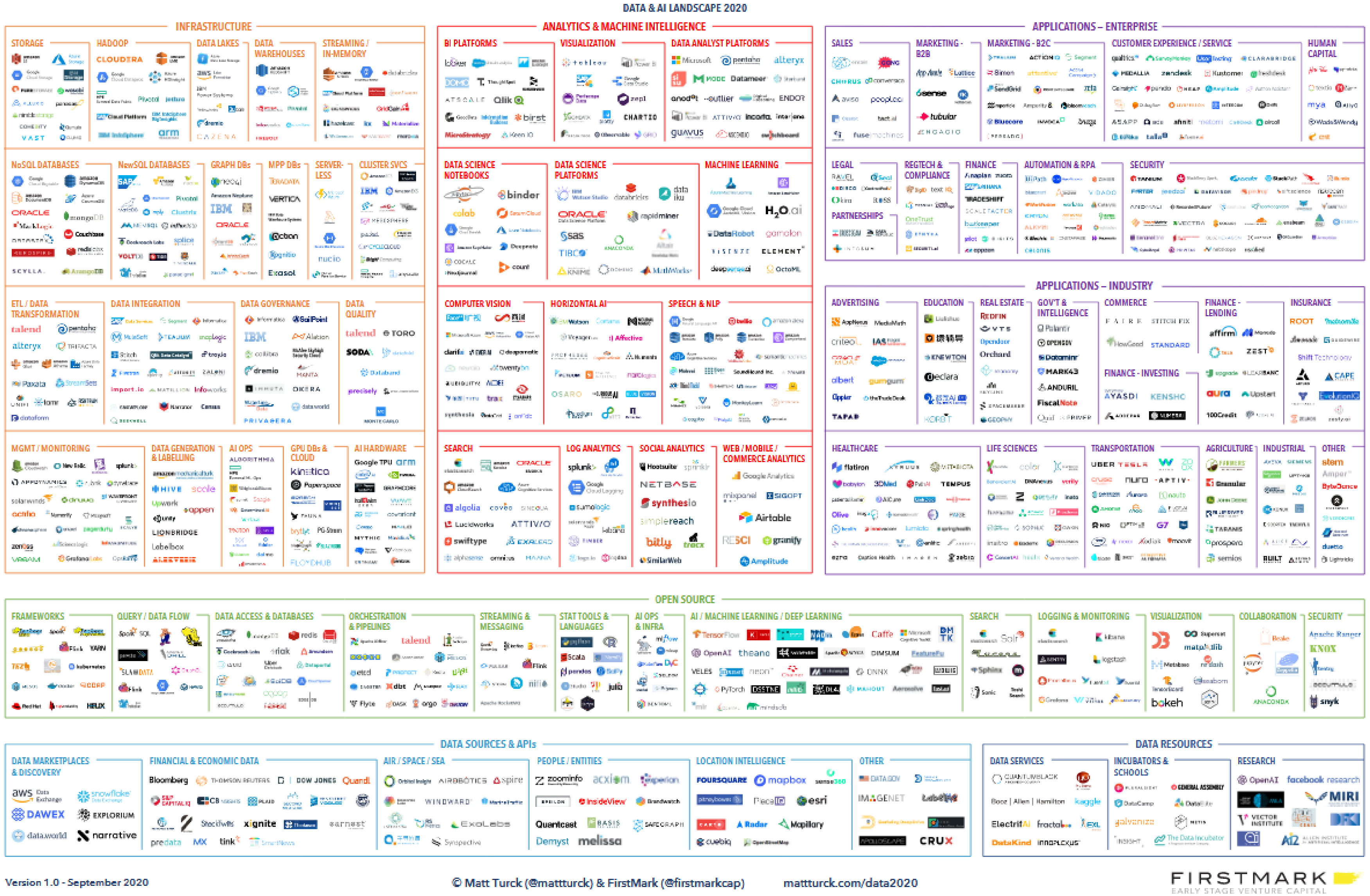
- Resources
  - Reviews Data – sentiments.csv

- Materials
  - Jupyter Notebook (Lab 10_2)
  - Several Python scripts and text files

Cluster Computing

- Introduction
- Hadoop
- Spark
- PySpark
- Amazon EMR

The Data & AI Landscape is vast!



https://mattturck.com/data2020/

Introduction

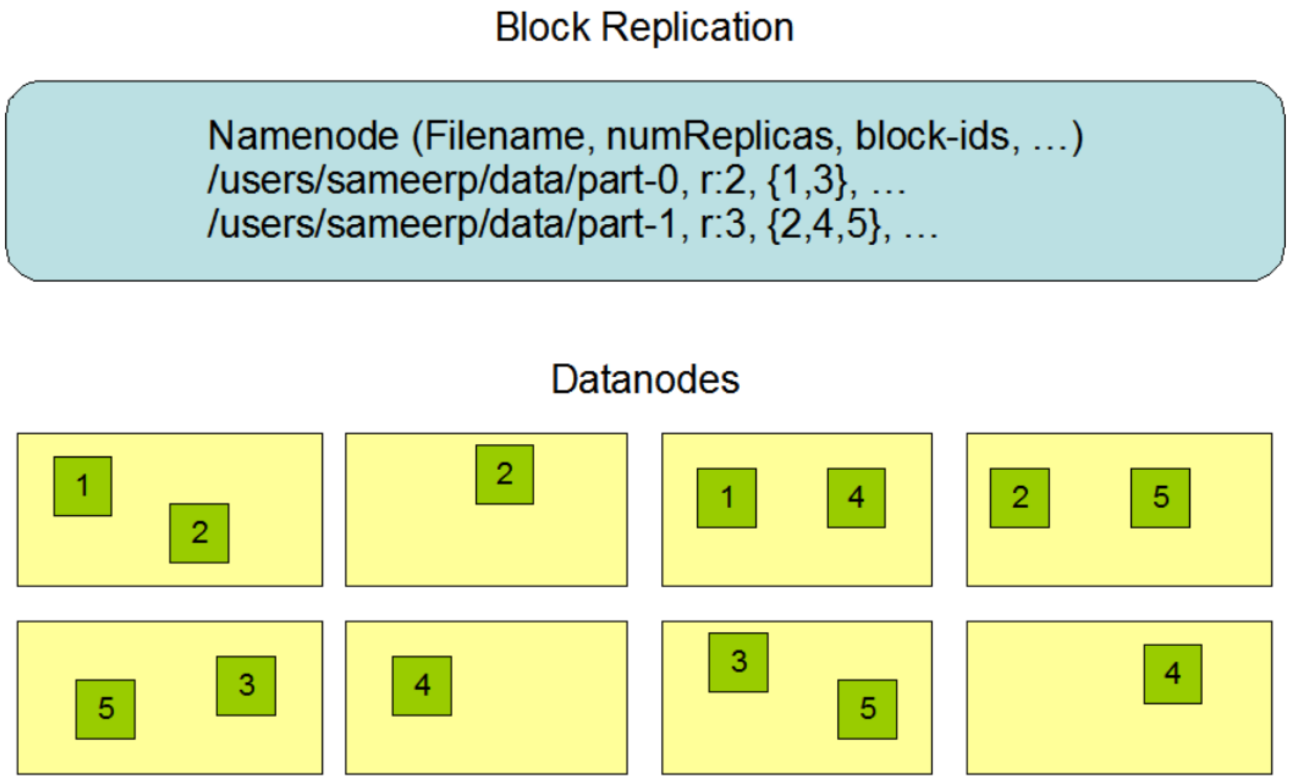
- **Cluster computing** – two or more computing nodes (servers) connected via a Fast Area Network, usually performing the same task
- Chief benefits – speed, scalability, flexibility over traditional mainframe solutions
- A **load balancer** may be used to spread the computing load evenly across the cluster for better overall performance
- Applications in Machine Learning
  - Working with large datasets
  - Deep learning (often nodes will have GPUs or TPUs)

Parallel Computation

- MapReduce: splits a large task into multiple map and reduce tasks that can be distributed across a cluster
  - **Map**: Process chunks (splits) of data in parallel creating intermediate results as key-value pairs
  - **Shuffling**: Pairs with the same key are sent to the same reducer
  - **Reduce**: Process in parallel each of the pairs having the same key
- Hadoop: Apache's implementation of MapReduce
- YARN: Performs resource management and job scheduling for Hadoop
- Apache Spark: For in-memory data processing

HDFS – Hadoop Distributed File System (Big Data Storage)

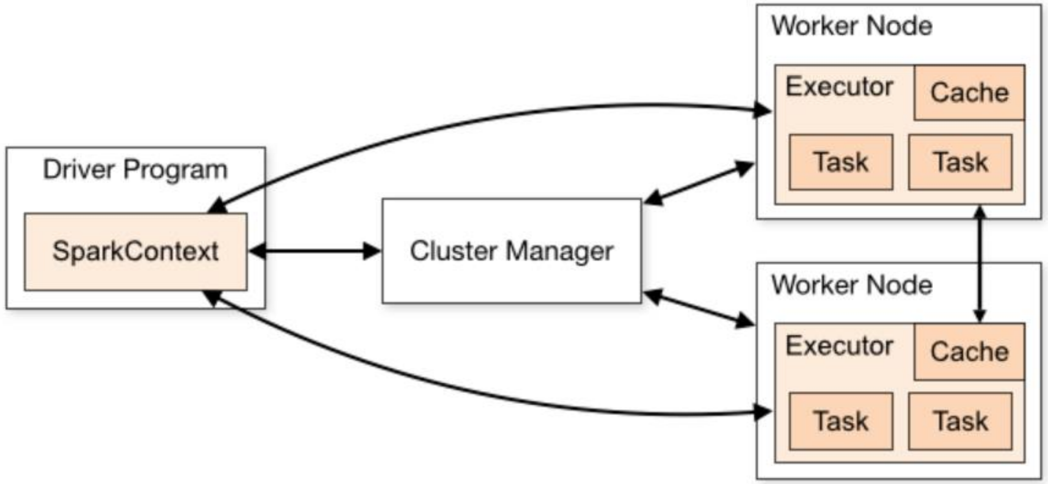
- Inputs and outputs of a map-reduce operation are stored in HDFS
- **Namenode** is a master node that contains metadata about how data is to be distributed in the cluster
- Here a single large file is split into five parts indicated by block-ids 1 to 5. Blocks 1 and 3 have two copies while 2, 4, 5 are replicated three times among the Datanodes
- Replication of data ensures fault-tolerance (default replication value is 3 in Hadoop)



https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Spark and PySpark

- Spark accesses data from HDFS but avoids MapReduce processing, instead performing in-memory computing instead of on disk.
  - hence a faster alternative
- PySpark is a Spark API for Python
- A SparkContext represents the connection to a Spark cluster
- Subpackages:
  - pyspark.sql module
  - pyspark.streaming module
  - pyspark.ml package
  - pyspark.resource module



https://spark.apache.org/docs/latest/cluster-overview.html

pyspark.sql

- This provides SQL-like functions for Spark DataFrames
    - SparkSession – create this to start working with a DataFrame
    - createDataFrame – via a variety of file formats such as csv, JSON, ORC, Parquet, Hive tables, Avro

```
data = [(‘John','','Smith',’1993-03-06','M’,4000),
  (‘Gareth’,’Smart','',’2002-07-29',’M’,5000),
  (‘Linda',“,’Carey',’1988-05-06',’F’,5000),
]

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName(‘Name of App').getOrCreate()
columns = ["firstname","middlename","lastname","dob", "gender","salary"]
df = spark.createDataFrame(data=data, schema = columns)
```

pyspark.sql

Note that Spark Dataframes are immutable, so any modification needs to be written to a new dataframe

```
#update a column
dfnew = df.withColumn("salary",col("salary")*100)

#show all rows
dfnew.show(truncate=False)

#use select for column subsetting, filter for row subsetting
dfnew.select("salary")
dfnew.filter(dfnew."salary" > 2000)
```

Big Data – Example tools

- Hue, Apache Impala – real-time SQL for data warehouses (the latter for Hadoop)
- Pig/Hive: SQL-like scripting and querying tools for data processing that simplify MapReduce programs.
- HBase, MongoDB, Elasticsearch: Examples of a few NoSQL databases.
- Mahout, Spark ML: Tools for running scalable machine learning algorithms in a distributed fashion.
- Flume, Sqoop, Logstash: Data ingestion and integration (e.g. transfer of data between databases)
- Splunk, Elastic Stack: Monitoring of log files

Amazon EMR

- Amazon Elastic Map Reduce – distributes processing over EC2 clusters
- Master node – manages the cluster
- Core nodes – run tasks and store results in HDFS
- Task nods (optional) – run computation but do not make use of HDFS storage
- Spark Livy – a web service that connects with Spark over a REST interface enabling one to connect SageMaker to a cluster

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	**m4.large** 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	**On-demand** **Spot** Use on-demand as max price
Core Core - 2	**m4.large** 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	2 Instances	**On-demand** **Spot** Use on-demand as max price
Task Task - 3	**m5.xlarge** 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB Add configuration settings	0 Instances	**On-demand** **Spot** Use on-demand as max price

Lab 10.3: Big Data Computing

- Purpose
  - Learn how to use PySpark to analyse a large dataset
- Resources
  - A population dataset from Population Division, United Nations
- Materials
  - Jupyter Notebook (Lab 10_3) in Google Colab

Institute of Data

Stream Computing

- Data Stream Processing
- The Pub/Sub Model
- Apache Kafka
- Amazon Kinesis
- Amazon Kinesis Analytics (SQL)
    - Windowed queries

Data Streams are everywhere

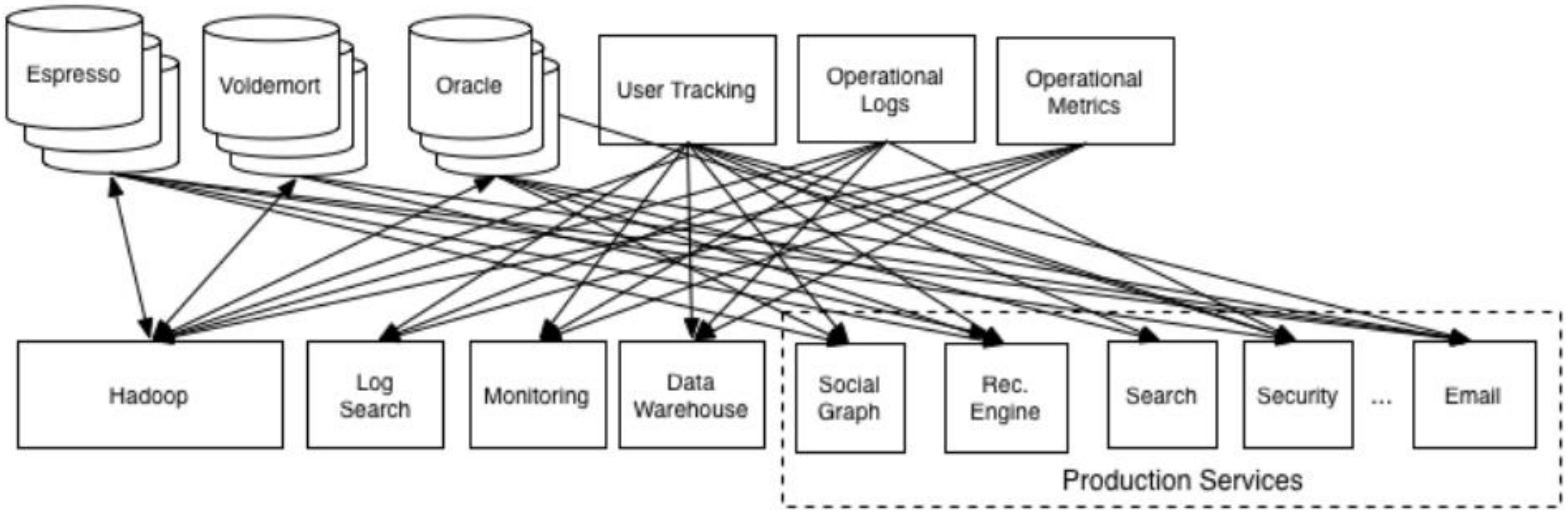
- Sensor data
- Video
- News feeds
- Financial data
- Online gaming
- Network Traffic data
- Clickstreams or App activity tracking
- Social media
- Location tracking
- Machine logs

Data Stream Processing

- Typically a solution comes from the use of multiple tools
- A stream process can be thought of as a directed acyclic graph composed of **sources**, **operators** and **sinks**
- **Sources** – where streams enter the streaming system
- **Operators** – functions of streams producing an output stream
- **Sinks** – places where streams flow out of the streaming system (e.g. a dashboard or database)
- Sources, sinks, operators combined into directed acyclic graph
- No batch processing, outputs constantly produced
- Event time and processing time generally differ
- Computation performed upon every event or within time windows

The Pub/Sub Model

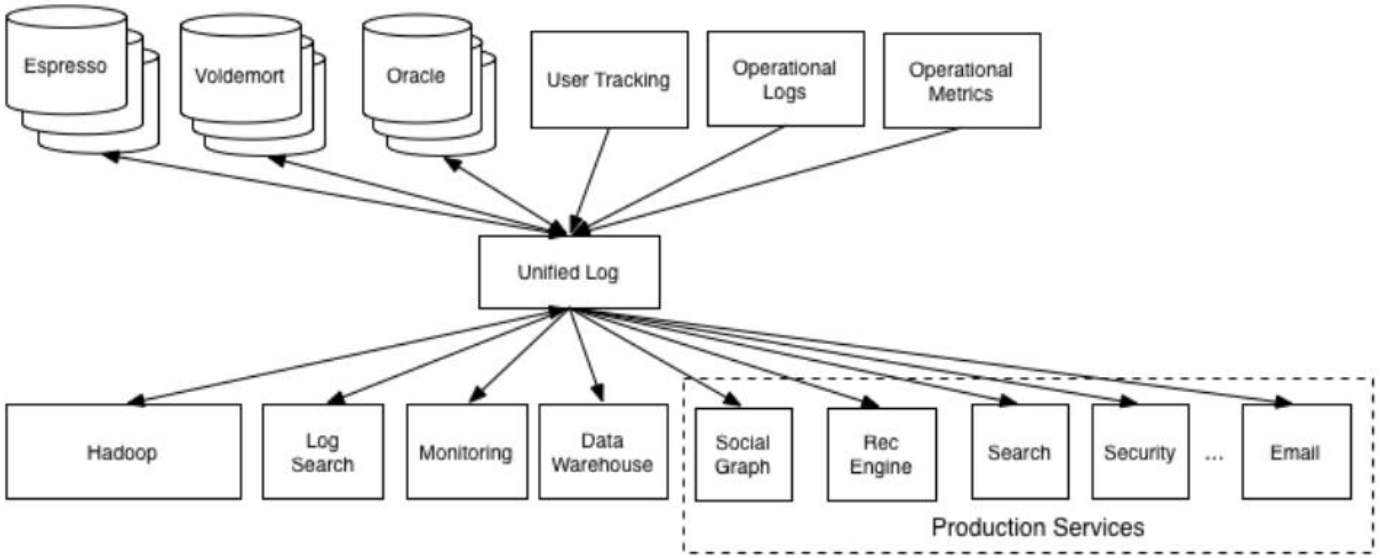
- Traditionally data systems and repositories are connected in the following manner, with a separate pipeline needed for each connection



https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

The Pub/Sub Model

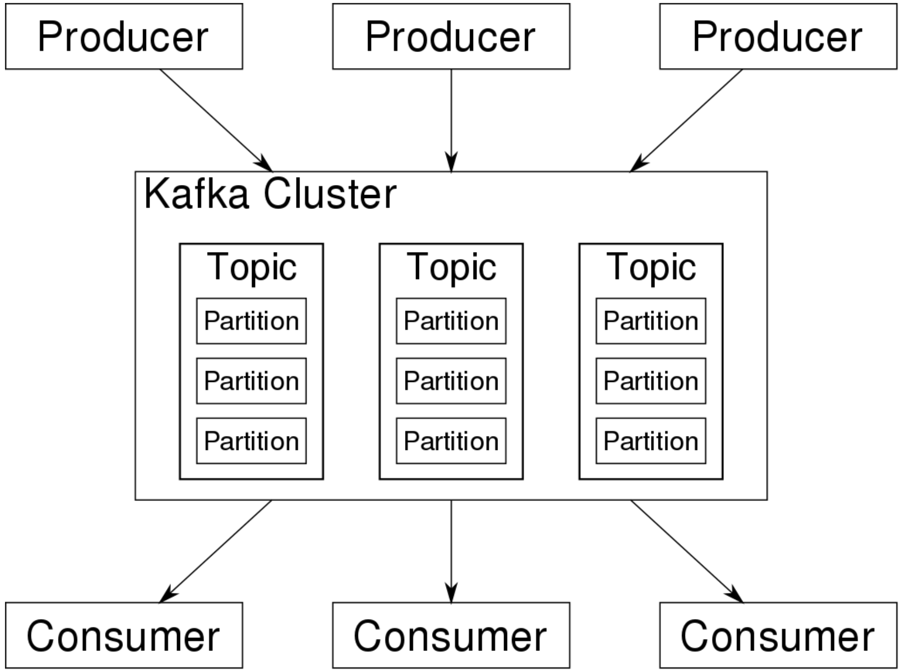
- Connections can be greatly reduced by separating the **producers** and **consumers** of data, each publishing to/subscribing from a messaging system
- **Apache Kafka** is one such model employing distributed stream processing



https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

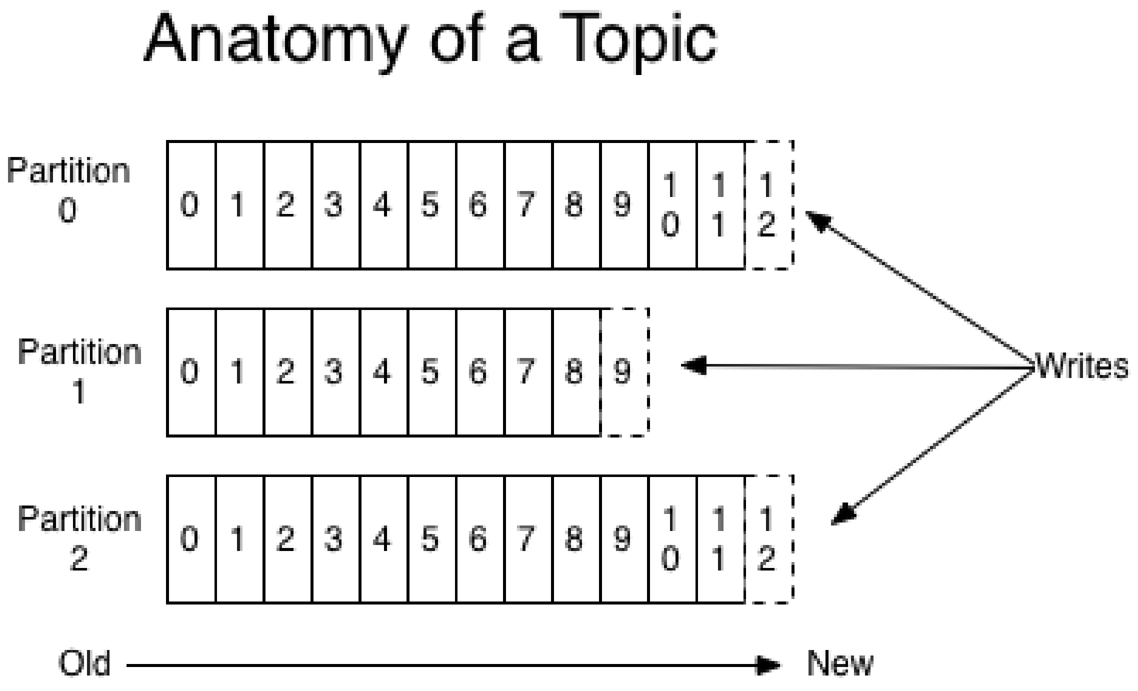
Apache Kafka

- **Broker** - handles all requests from clients (producers, consumers and metadata) and keeps data replicated within the cluster.
- **Topics** (categories) are divided into **partitions** containing records each with a unique offset in a fixed sequence. Partitions may be replicated across the cluster.



Apache Kafka

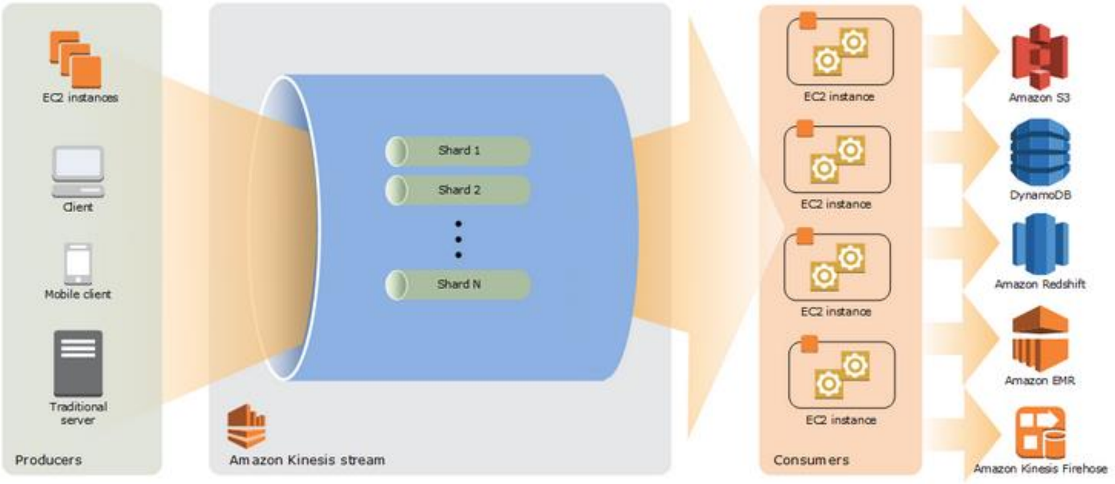
- **Message (record)** - a key / value pair – the key assigns records to a specific partition (the publisher can also specify a partition directly)
- A record inside a partition has an **offset**, used to identify the order in which records are published to a partition – hence messages can either be read from the beginning or a specified offset
- **Zookeeper** – used for service synchronisation, tracking the status of nodes (brokers) in the Kafka cluster and maintaining a list of Kafka topics and messages (metadata)



https://sookocheff.com/post/kafka/kafka-in-a-nutshell/log-anatomy.png

Amazon Kinesis

- AWS platform for streaming data
- The throughput is in units of **shards**
- To put data into the stream, one specifies the **name** of the stream, a **partition key**, and the **data blob** to be added to the stream
- The partition key is used to determine which shard in the stream the data record is added to.



- 1 shard can support up to 5 transactions per second for reads, up to a maximum total data read rate of 2 MB per second and up to 1,000 records per second for writes, up to a maximum total data write rate of 1 MB per second (typically consumers are slower in transactions than producers)

Lab 10.4 (Optional)

- Purpose
  - Learn how to use Apache Kafka in data streaming applications
- Resources
  - fraud_data.csv
- Materials
  - Jupyter Notebook (Lab 10_4) in Google Colab

Questions?

Appendices

End of Presentation!