AEROSPIKE

NEXTGEN NOW SUMMIT '20

Infrastructure for Machine Learning Applications

AEROSPIKE

Natalie Pistunovich
Lead Developer Advocate
AEROSPIKE

Hello

- About me
  - I'm Natalie Pistunovich (@NataliePis on Twitter)
  - Lead Developer Advocate at Aerospike
  - Google Developer Expert for Go
  - Previously: Software, Hardware and Infrastructure Engineer and Engineering Manager

Hello

- Our plan for the day
  - How to ML
  - Infrastructure for ML
  - Bare bones ML infrastructure
  - A hands on example
  - Demo
  - Recap
  - Q&A

What's ML?

Machine Learning

... is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.

It is seen as a subset of artificial intelligence.

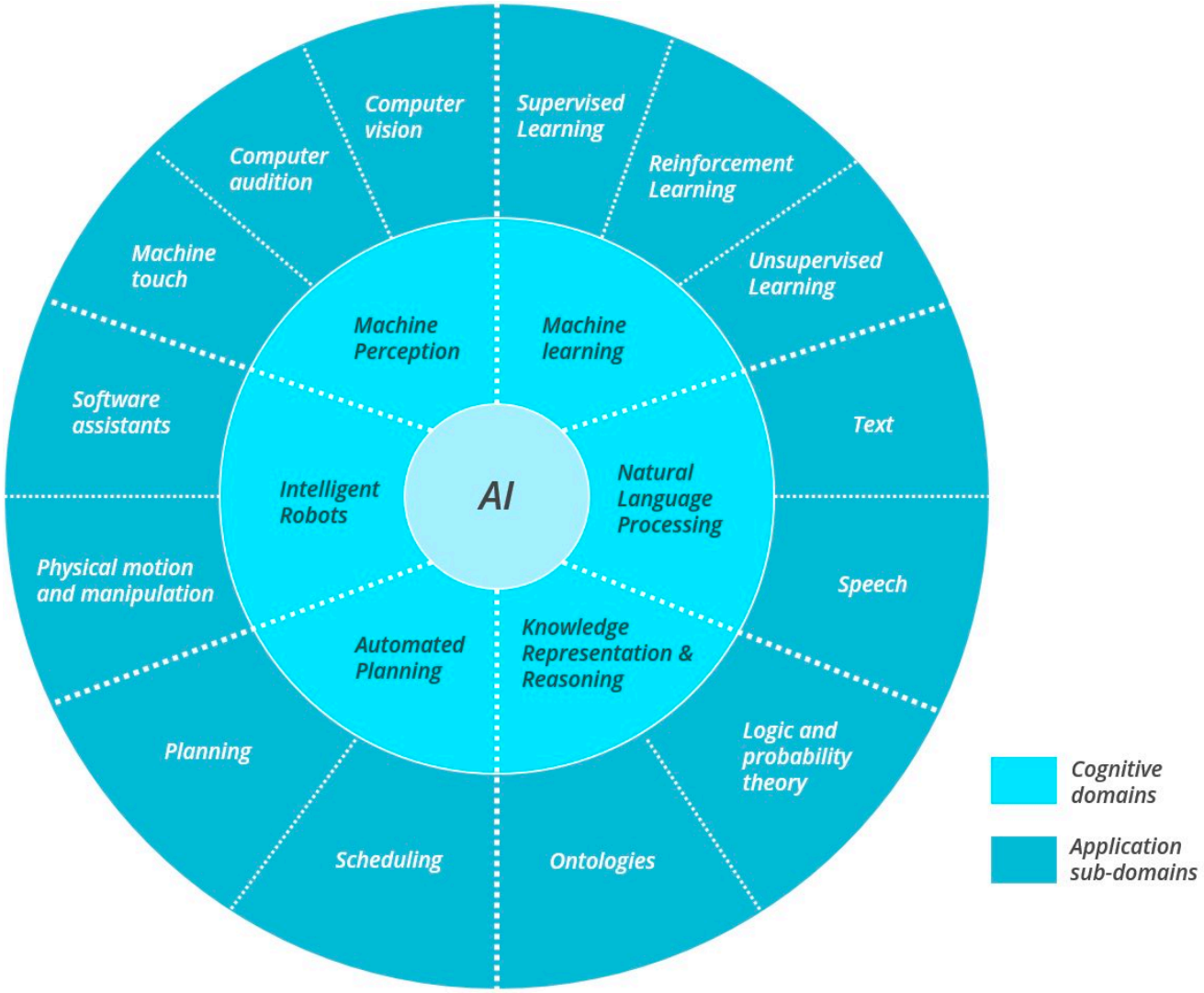
– Wikipedia

Artificial Intelligence

In computer science, **artificial intelligence** (**AI**) is the intelligence demonstrated by machines, in contrast to the **natural intelligence** displayed by humans.

– Wikipedia

Machine Learning



How to ML?

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. **Define the problem**
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. **Gather data**
    - Relevant to the problem
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. Gather data
3. **Prepare data**
   Clean, pre-process, randomize, split: train/test
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. **Choose a model**
    - Depending on: problem, input type, categories number
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. **Train the model**
    - Assign random values, predict, evaluate results, adjust values
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. **Evaluate the model**
    Using test data
7. Tune the hyperparameters
8. Predict

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. **Tune the hyperparameters**
8. Predict

How to ML?

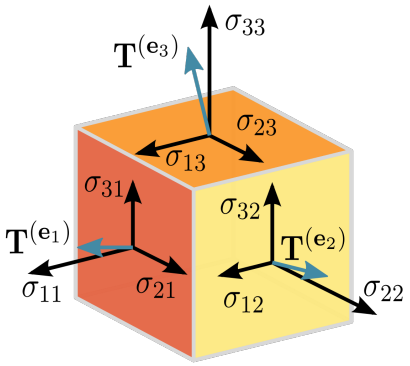
1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. **Predict**

How to ML?

TensorFlow is an open-source software for Machine Intelligence, used mainly for machine learning applications such as neural networks.

How to ML?

TensorFlow is an open-source software for Machine Intelligence, used mainly for machine learning applications such as neural networks.

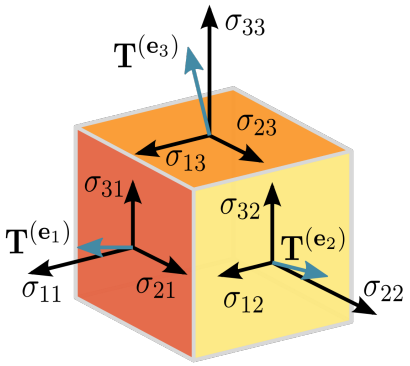


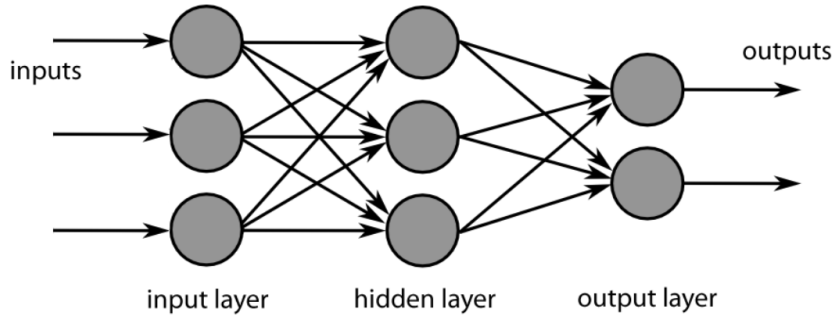
A tensor is a generalization of vectors and matrices to potentially higher dimensions

1. data type
2. shape
   - number of dimensions
   - number of values / dimension

How to ML?

TensorFlow is an open-source software for Machine Intelligence, used mainly for machine learning applications such as neural networks.





A tensor is a generalization of vectors and matrices to potentially higher dimensions

1. data type
2. shape
   - number of dimensions
   - number of values / dimension

The flow part comes to describe:
- the graph (model) is a set of nodes (operations)
- the data (tensors) "flows" through those nodes, undergoing mathematical manipulation

You can look at, and evaluate, any node of the graph

ML Infrastructure

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

Abstract

Machine learning offers a fantastically powerful toolkit for building useful complex prediction systems quickly. This paper argues it is dangerous to think of these quick wins as coming for free. Using the software engineering framework of *technical debt*, we find it is common to incur massive ongoing maintenance costs in real-world ML systems. We explore several ML-specific risk factors to account for in system design. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, configuration issues, changes in the external world, and a variety of system-level anti-patterns.

1 Introduction

As the machine learning (ML) community continues to accumulate years of experience with live systems, a wide-spread and uncomfortable trend has emerged: developing and deploying ML systems is relatively fast and cheap, but maintaining them over time is difficult and expensive.

Infrastructure

There's a lot more to Machine Learning than just implementing the ML algorithm.

Infrastructure

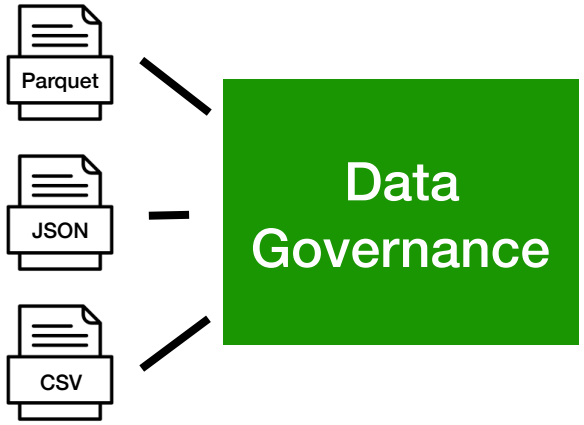
The ML code is at the heart of a real-world production system, but it accounts for **5% or less** of the overall code of that system.

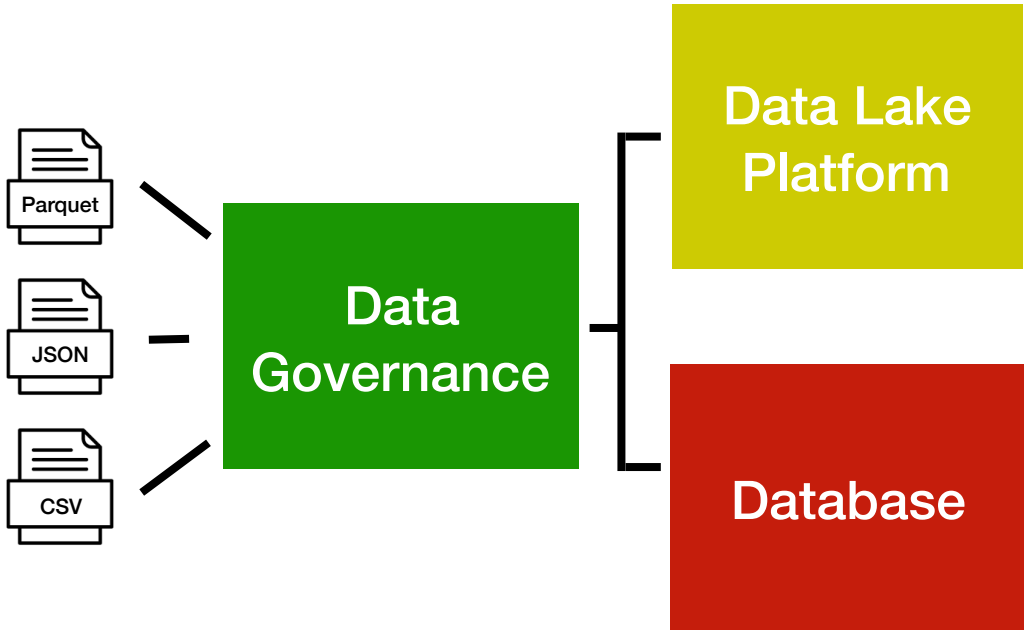
A Bare Bones Infrastructure

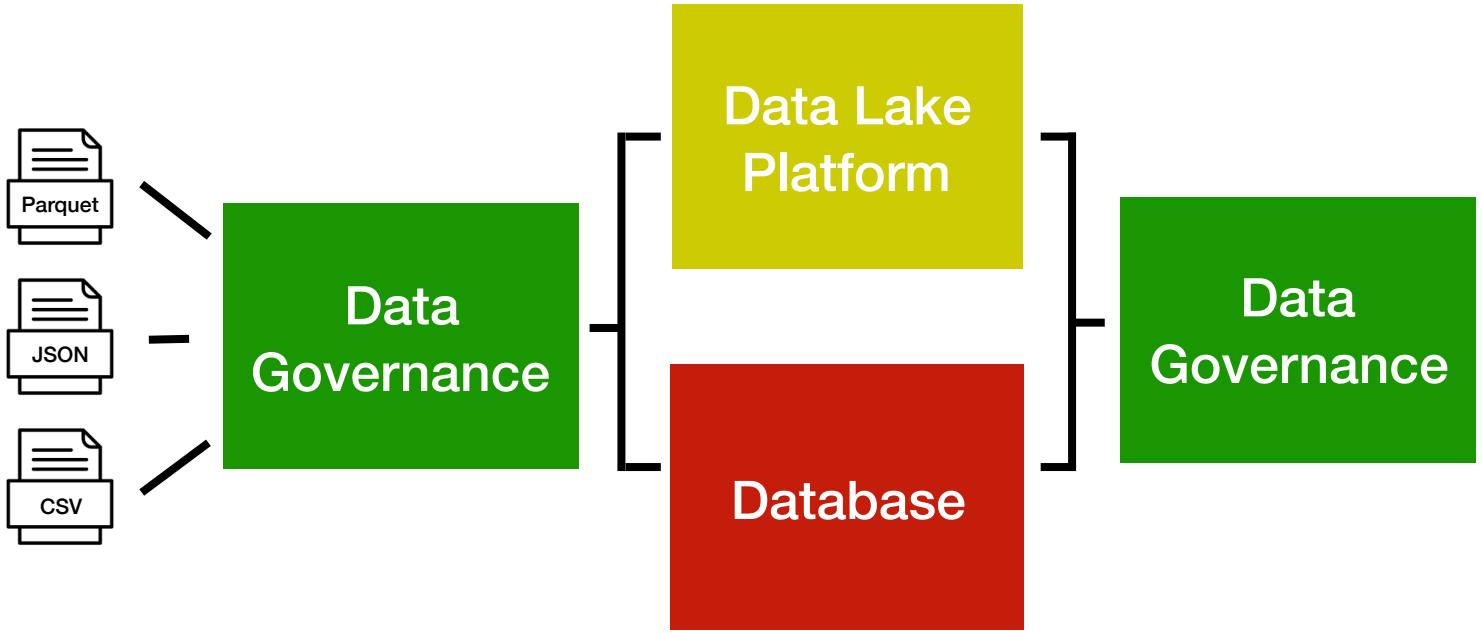

Parquet

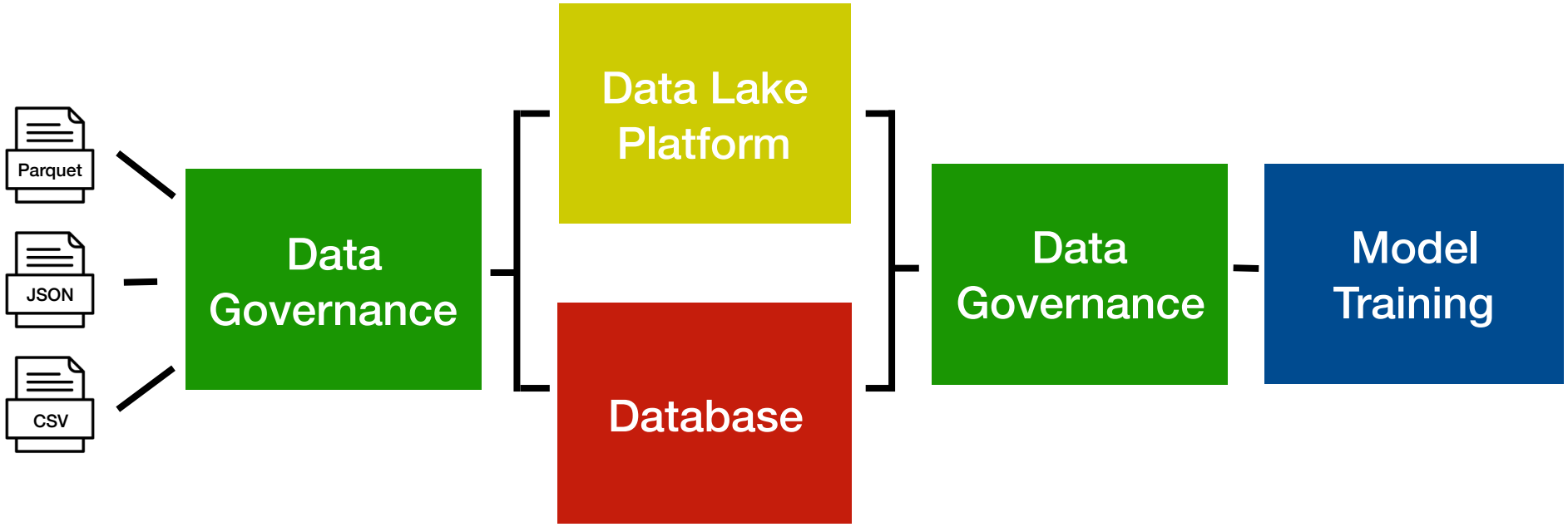

JSON


CSV


Parquet
JSON
CSV
Data
Governance


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. **Train the model**
    - Assign random values, predict, evaluate results, adjust values
6. Evaluate the model
7. Tune the hyperparameters
8. Predict

Types of Trained Models

Static model (Offline) vs. Dynamic model (Online)

Static Model

+	−
Easier to build and test.	Can only predict things we know about. Update latency likely measured in hours or days.

Dynamic Model

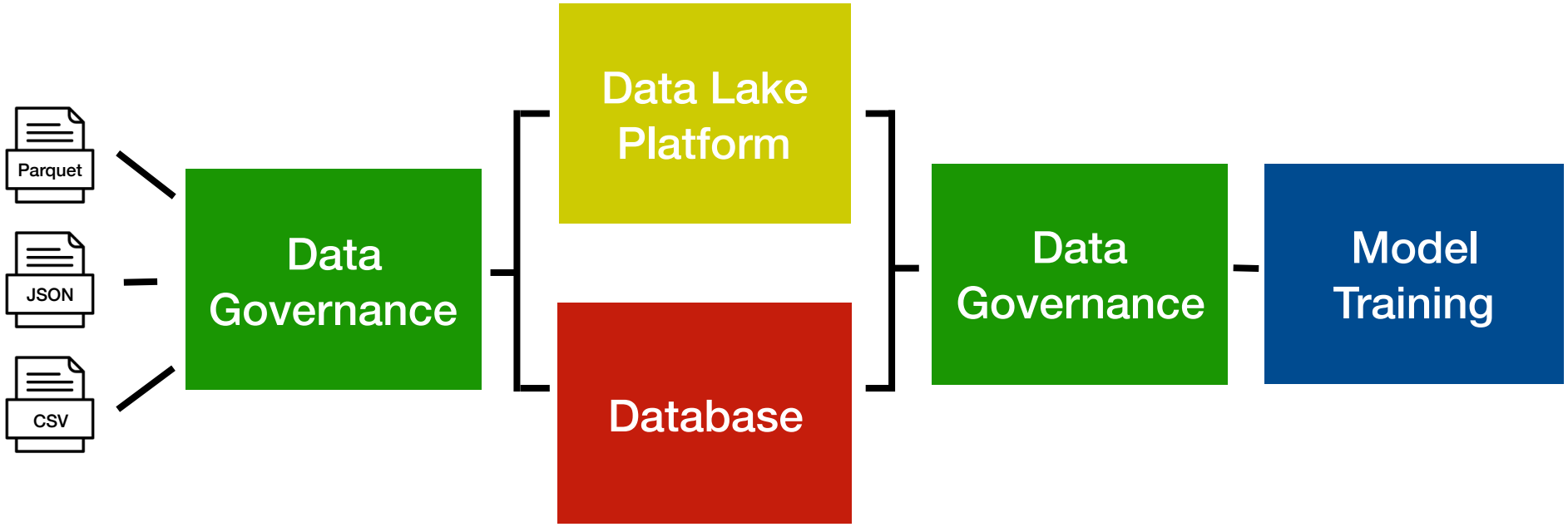
\+

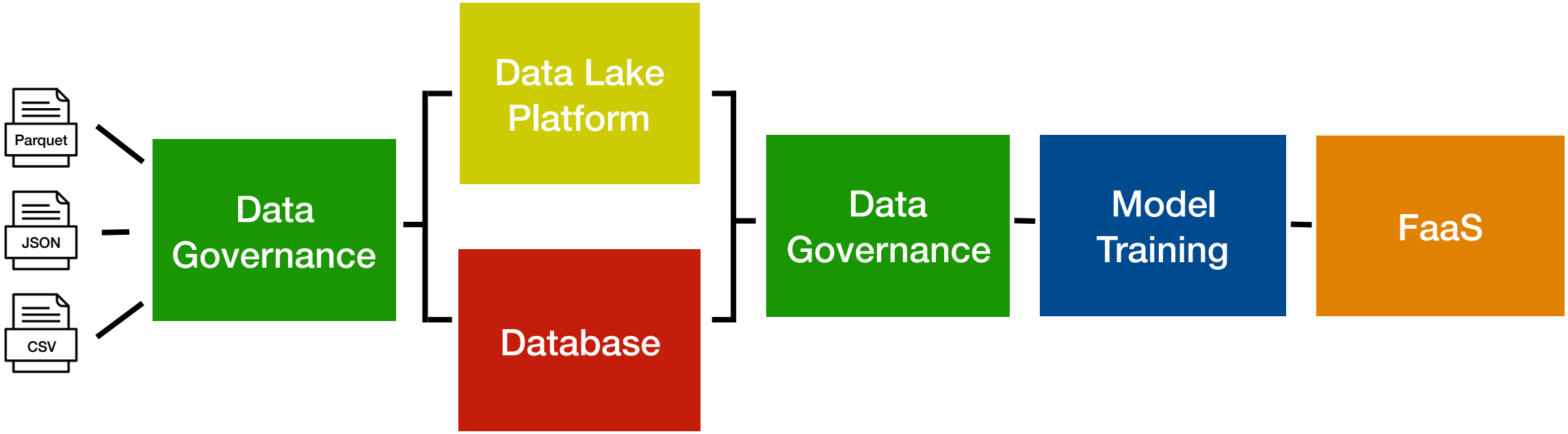
Adapted to changing data, hence more likely to make better predictions.

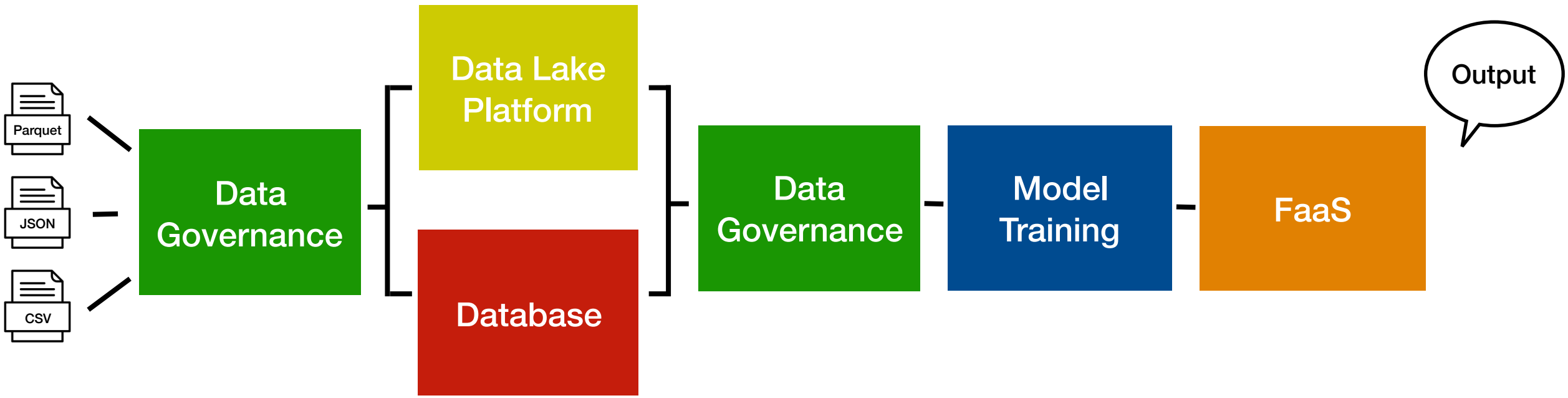
\-

Compute intensive, latency sensitive, may limit model complexity.

Monitoring needs are more intensive: outputs and performance.


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training
FaaS


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training
FaaS
Output

How to ML?

1. Define the problem
2. Gather data
3. Prepare data
4. Choose a model
5. Train the model
6. Evaluate the model
7. Tune the hyperparameters
8. **Predict**

Types of Inference

Online Inference vs. Offline Inference

Online Inference

\+

Can make a prediction on any new item as it comes in — great for long tail.

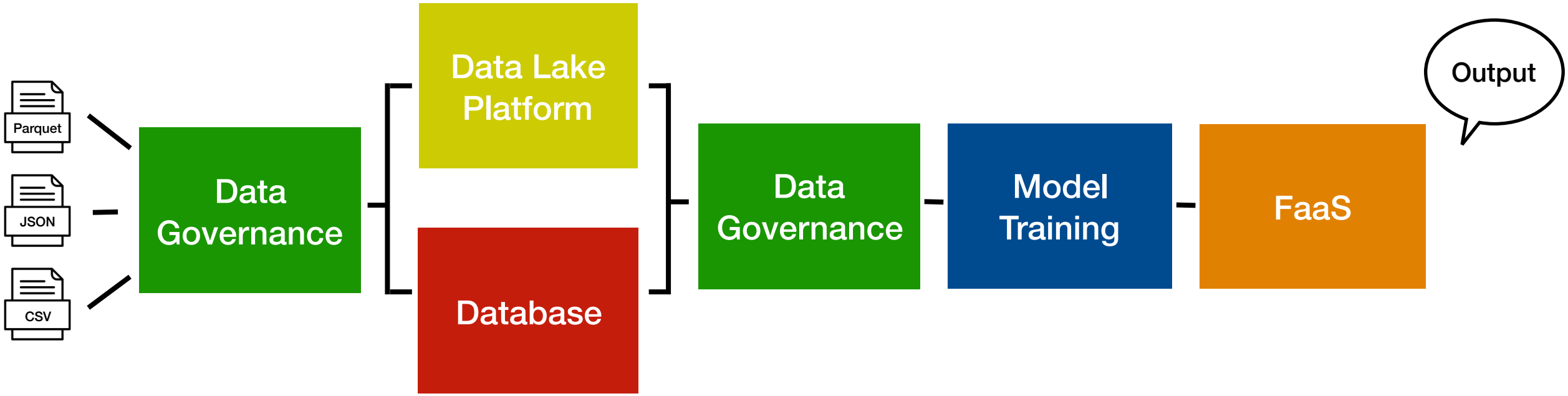
\-

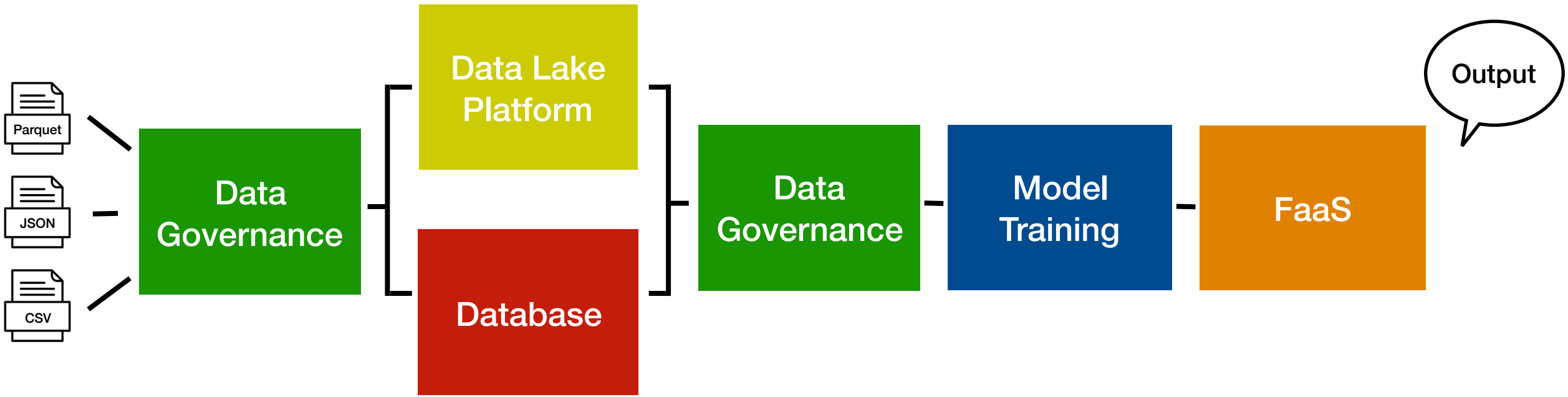
Compute intensive, latency sensitive — may limit model complexity.

Monitoring needs are more intensive.

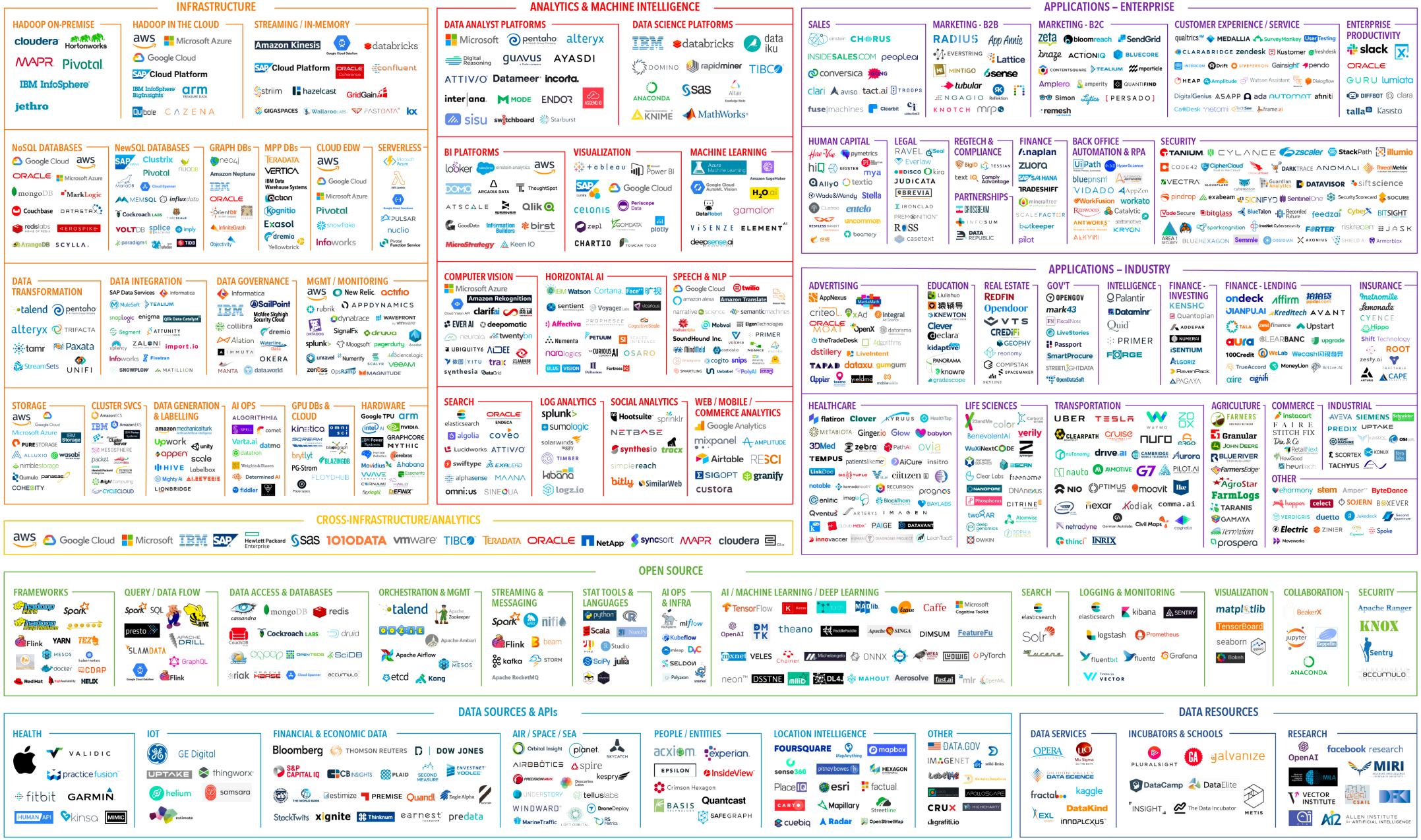
Offline Inference

+	−
Predictable low cost.	Can only predict based on known information.
Likely to use batch quota.	
All predictions can be verified before going live.	

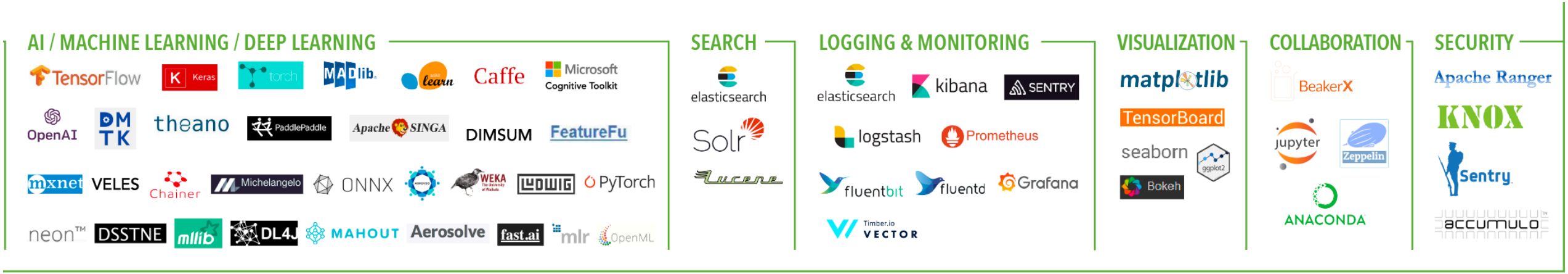

Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training
FaaS
Output


Parquet
JSON
CSV
Data Governance
Data Lake Platform
Database
Data Governance
Model Training
FaaS
Output

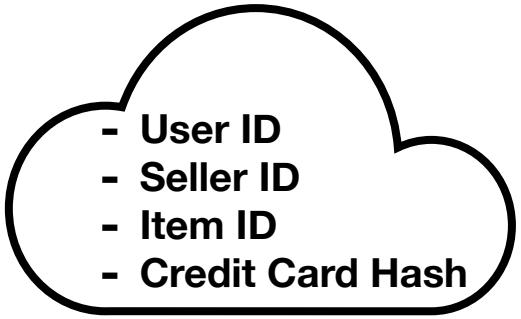
Monitoring

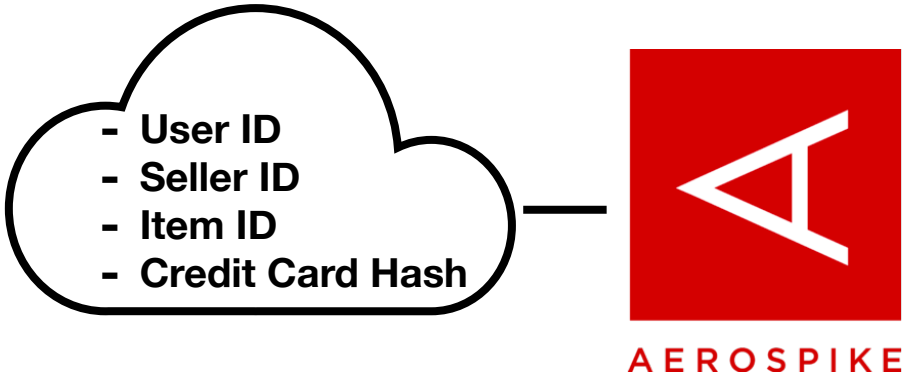



HADOOP ON-PREMISE
cloudera
Hortonworks
MAPR
Pivotal
IBM InfoSphere
jethro
HADOOP IN THE CLOUD
aws
Microsoft Azure
Google Cloud
SAP Cloud Platform
IBM InfoSphere BigInsights
arm TREASURE DATA
Qubole
CAZENA
STREAMING / IN-MEMORY
Amazon Kinesis
Google Cloud Dataflow
databricks
SAP Cloud Platform
ORACLE Coherence
confluent
striim
hazelcast
GridGain
GIGASPACES
Wallaroo LABS
FASTDATA
kx
NoSQL DATABASES
Google Cloud
aws
ORACLE
Microsoft Azure
mongoDB
MarkLogic
Couchbase
DATASTAX
redislabs
EROSPIKE
ArangoDB
SCYLLA.
NewSQL DATABASES
SAP HANA
Clustrix
NUODB
Pivotal
MariaDB
Cloud Spanner
MEMSQL
influxdata
Cockroach LABS
TIMESCALE
VOLTDB
splice MACHINE
imply
paradigm4
Trafodion
TiDB
GRAPH DBs
neo4j
Amazon Neptune
IBM
ORACLE
OrientDB
InfiniteGraph
Objectivity
MPP DBs
TERADATA
VERTICA
IBM Data Warehouse Systems
actian
Kognitio
Exasol
dremio
Yellowbrick
CLOUD EDW
aws
Google Cloud
Microsoft Azure
Pivotal
snowflake
Infoworks
SERVERLESS
Microsoft Azure
AWS Lambda
Google Cloud Functions
PULSAR
nuclio
Pivotal Function Service
DATA TRANSFORMATION
talend
pentaho
alteryx
TRIFACTA
tamr
Paxata
StreamSets
UNIFI
DATA INTEGRATION
SAP Data Services
Informatica
MuleSoft
TEALIUM
snaplogic
enigma
Qlik Data Catalyst
Segment
ATTUNITY
xplenty
ZALONI
import.io
Infoworks
Fivetran
SNOWPLOW
MATILLION
DATA GOVERNANCE
Informatica
IBM
SailPoint
McAfee Skyhigh Security Cloud
collibra
dremio
Alation
Waterline Data
IMMUTA
OKERA
MANTA
data.world
MGMT / MONITORING
aws
New Relic.
actifio
rubrik
APPDYNAMICS
dynatrace
WAVEFRONT by vmware
DATADOG
SignalFx
druva
splunk>
Moogsoft
pagerduty
Anodot
unravel
Numerify
SCALYR
ScienceLogic
zenoss
OpsRamp
MAGNITUDE
VEEAM
STORAGE
aws
Google Cloud
Microsoft Azure
IBM Storage
PURESTORAGE
ALLUXIO
wasabi
nimblestorage
Qumulo
panasas
COHESITY
CLUSTER SVCS
Amazon ECS
IBM
Amazon EKS
IBM Power Systems
Microsoft Cluster Server
MESOSPHERE
packet
pepperdata
Hewlett Packard Enterprise
PENGUIN COMPUTING
Bright Computing
CYCLECLOUD
DATA GENERATION & LABELLING
amazon mechanical turk
Artificial Artificial Intelligence
Upwork
unity
appen
scale
HIVE
Labelbox
Mighty AI
LIONBRIDGE
AI OPS
ALGORITHMIA
SPELL
comet
Verta.ai
datmo
datatron
Weights & Biases
Determined AI
fiddler
GPU DBs & CLOUD
kinetica
omnisci
SQREAM TECHNOLOGIES
blazegraph
brytlyt
BLAZINGDB
PG-Strom
FLOYDHUB
Paperspace
HARDWARE
Google TPU
arm
intel AI
NVIDIA
IBM Power Systems
GRAPHCORE
MYTHIC
Horizon Robotics
cerebras
Movidius
habana
WAVE COMPUTING
Esperanto TECHNOLOGIES
CORNAMI
HAILO
flexlogix
EFINIX


AI / MACHINE LEARNING / DEEP LEARNING
TensorFlow
Keras
torch
MADlib
learn
Caffe
Microsoft Cognitive Toolkit
OpenAI
DMTK
theano
PaddlePaddle
Apache SINGA
DIMSUM
FeatureFu
mxnet
VELES
Chainer
Michelangelo
ONNX
WEKA
LUDWIG
PyTorch
neon™
DSSTNE
mllib
DL4J
MAHOUT
Aerosolve
fast.ai
mlr
OpenML
SEARCH
elasticsearch
Solr
Lucene
LOGGING & MONITORING
elasticsearch
kibana
SENTRY
logstash
Prometheus
fluentbit
fluentd
Grafana
Timber.io
VECTOR
VISUALIZATION
matplotlib
TensorBoard
seaborn
ggplot2
Bokeh
COLLABORATION
BeakerX
jupyter
Zeppelin
ANACONDA
SECURITY
Apache Ranger
KNOX
Sentry
accumulo

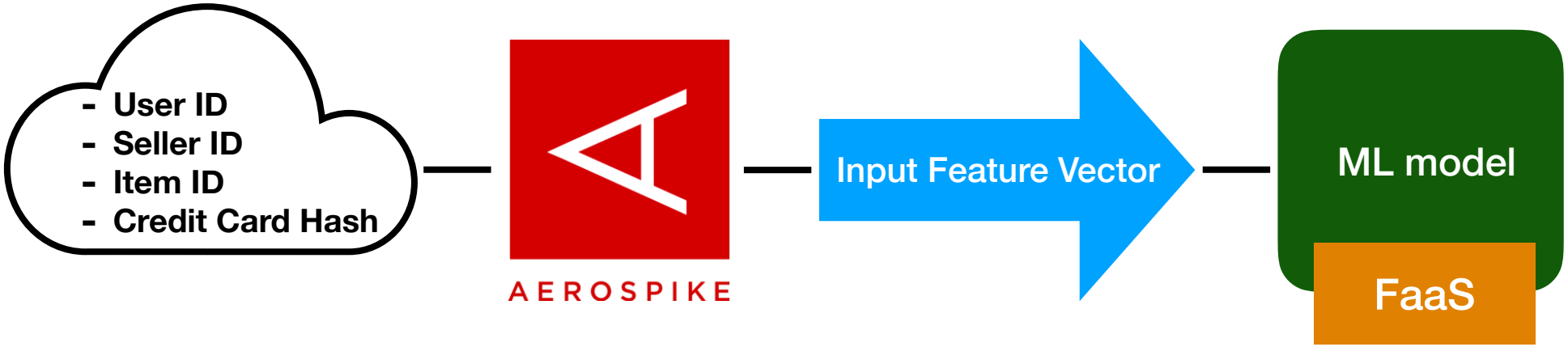
A Concrete Infrastructure Example

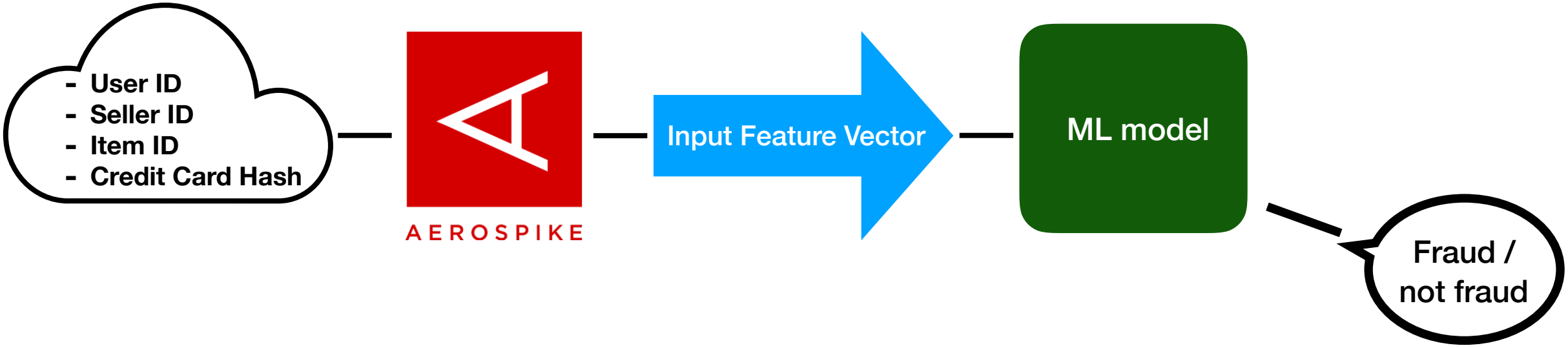

- User ID
- Seller ID
- Item ID
- Credit Card Hash


- User ID
- Seller ID
- Item ID
- Credit Card Hash
AEROSPIKE

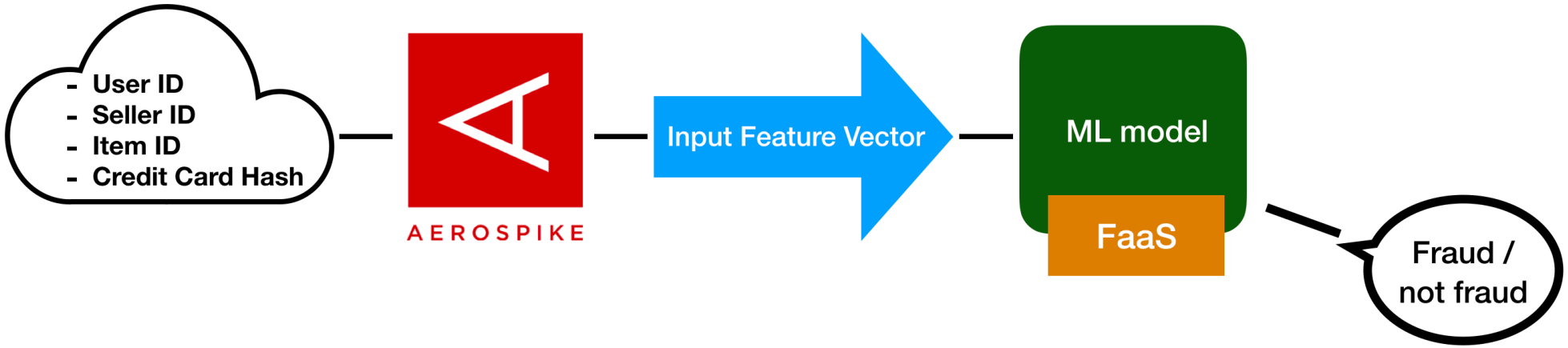
How long has the User been registered	1
How long has the Seller been registered	0
How many of all the user purchase were fraud?	0.7
How average is the purchase price for the User?	0.78
How often was the CC reported?	0
How often were the recent purchases with CC?	0.9
How standard is the num of items per Seller?	0.61


- User ID
- Seller ID
- Item ID
- Credit Card Hash
AEROSPIKE
Input Feature Vector


- User ID
- Seller ID
- Item ID
- Credit Card Hash
AEROSPIKE
Input Feature Vector
ML model
FaaS


- User ID
- Seller ID
- Item ID
- Credit Card Hash
AEROSPIKE
Input Feature Vector
ML model
Fraud /
not fraud

Demo Time!


- User ID
- Seller ID
- Item ID
- Credit Card Hash
AEROSPIKE
Input Feature Vector
ML model
FaaS
Fraud / not fraud

Recap

Recap

- What's AI? What's ML? How to ML?

Recap

- What's AI? What's ML? How to ML?
- Training and Inference: Online vs. Offline

Recap

- What's AI? What's ML? How to ML?
- Training and Inference: Online vs. Offline
- Bare bones architecture components

Recap

- What's AI? What's ML? How to ML?
- Training and Inference: Online vs. Offline
- Bare bones architecture components
- A concrete architecture example

Recap

- What's AI? What's ML? How to ML?
- Training and Inference: Online vs. Offline
- Bare bones architecture components
- A concrete architecture example

Recap

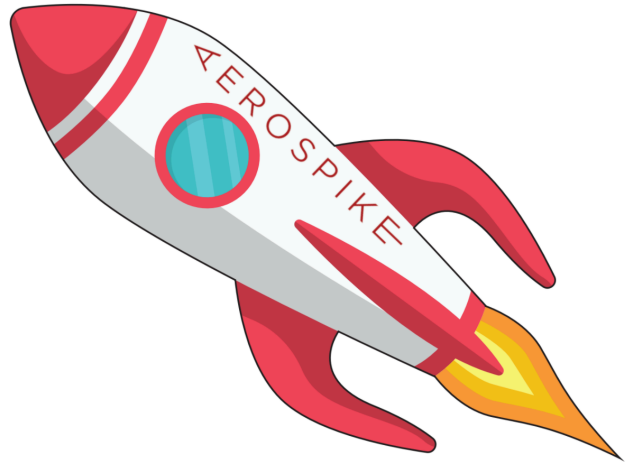
- ML code is about **5% or less** of the overall code of a ML system
- Input data is a big and tricky part of an ML system
- Bare bones flow:

process data → store data → train model →

→ serve predictions → monitor

Thank you!

@NataliePis @AerospikeDB
#NextGenNow #AS20



natalie@aerospike.com

tinyurl.com/join-aerospike-slack