Data Analysis with Hadoop & Spark

Part One

Agenda

- Introduction to Distributed Computing
  - The Age of Data Products
  - Building Data Products at Scale
  - Data Product Architectures
- Hadoop: An Operating System for Big Data
  - Hadoop Architecture
  - What's In A Cluster?
  - HDFS Caveats
- Hands-on Lab

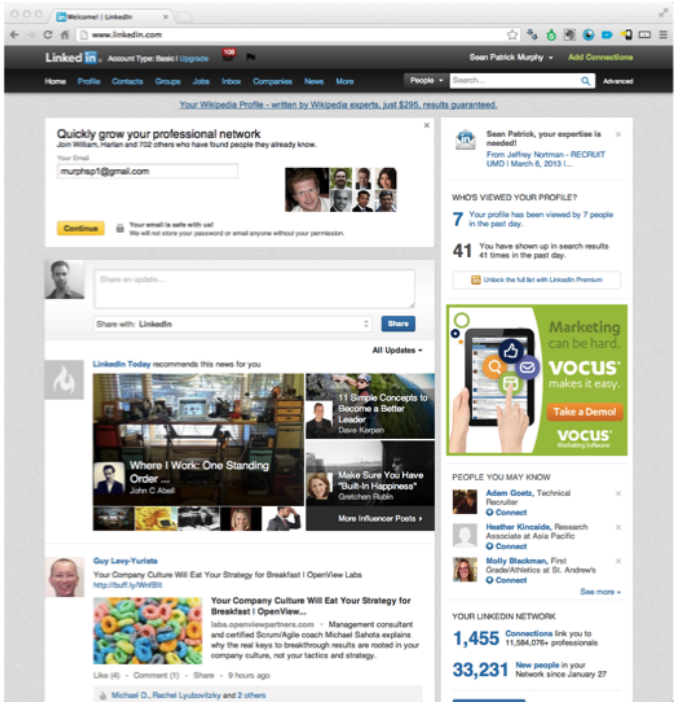
Introduction to Distributed Computing

The Age of Data Products

What is a data product?

"A data product is a product that is based on the combination of data and algorithms."

-- Hilary Mason


Quickly grow your professional network
Continue
Share
Marketing can be hard.
VOCUS makes it easy.
Take a Demo!
Where I Work: One Standing Order ...
Make Sure You Have "Built-In Happiness"
Your Company Culture Will Eat Your Strategy for Breakfast | OpenView...
1,455 Connections link you to 11,584,076+ professionals
33,231 New people in your Network since January 27

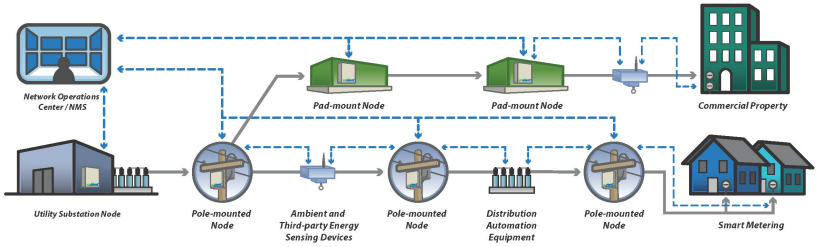

Google
Google Search
I'm Feeling Lucky

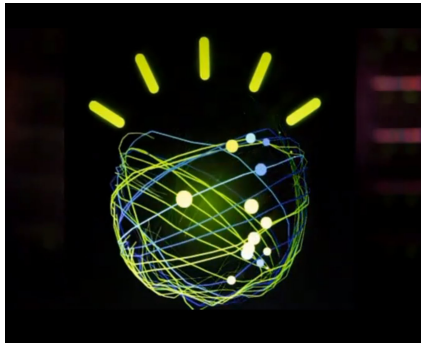

Carrier
3:19 PM
flights
Search Results
San Francisco → Hong Kong, 3/11/13
Depart:
Arrive:
12PM 12AM 12PM 12AM
3AM 3PM 3AM 3PM
$1,017
Singapore
$1,048
United
$1,030
Air Canada
$1,067
China Air
$1,148
Cathay*
$1,055
Korean
$1,068
ANA*
$1,068
United
$1,068
ANA*
$1,079
Air China
Agony
Price
Depart
Length

"A data application acquires its value from the data itself, and creates more data as a result. It's not just an application with data; it's a data product.

Data science enables the creation of data products."

-- Mike Loukides





Machine Learning for Big Data

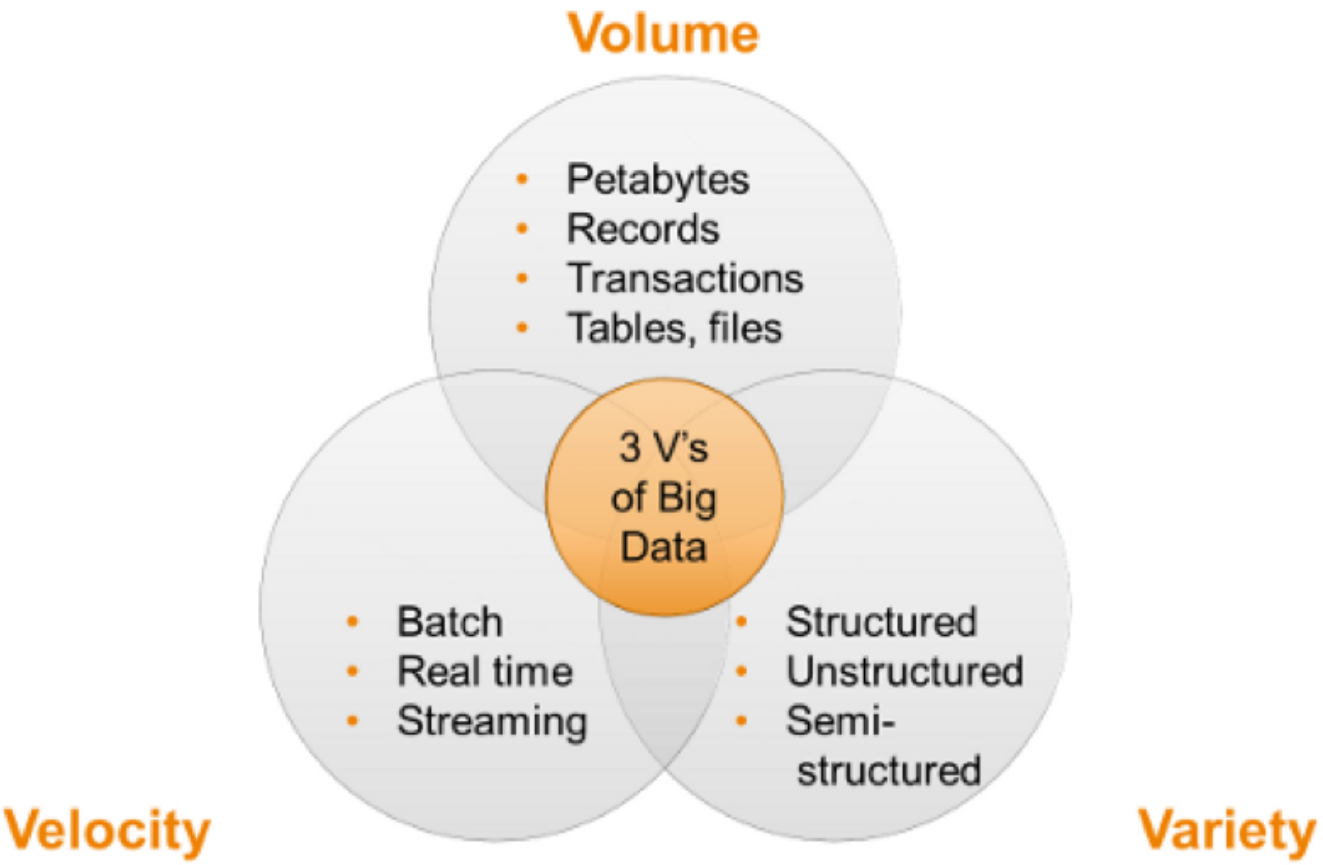
DistrictDataLabs

Data Products are self-adapting, broadly applicable software-based engines that derive their value from data and generate more data by influencing human behavior or by making inferences or predictions upon new data.

- District Data Labs

Building Data Products at Scale

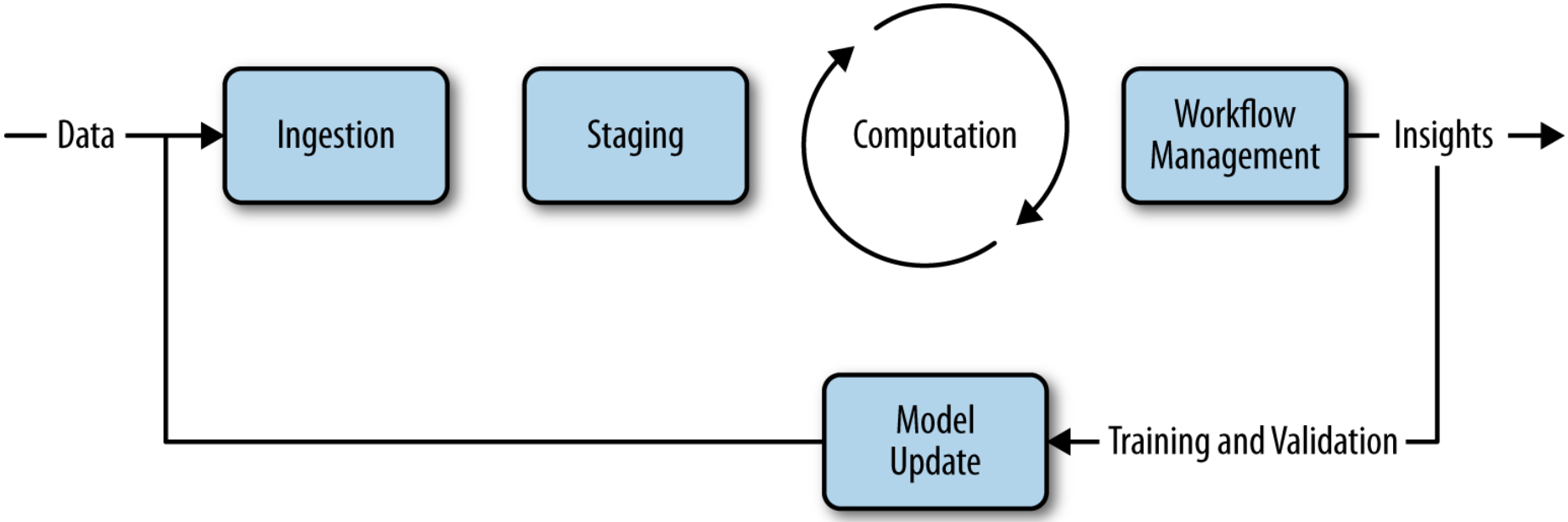
What Makes Data Big?



Possibilities for Big Data

- Higher dimensionality → increased complexity
- Solution: Machine learning (ML)
- ML learns by example → need lots of examples!

Big Data Pipeline



Let's Build a Data Machine

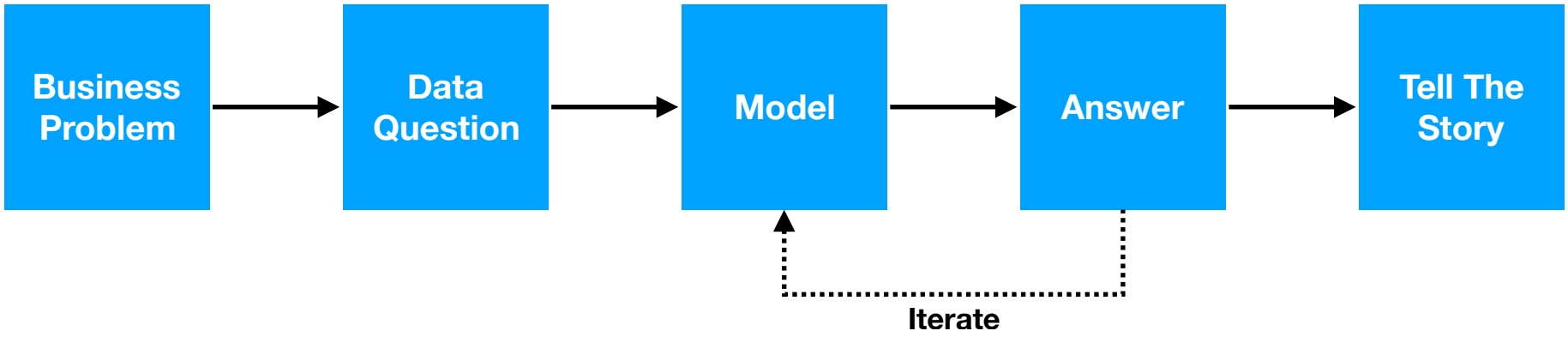
Input [ ] Output

Data Product Architectures

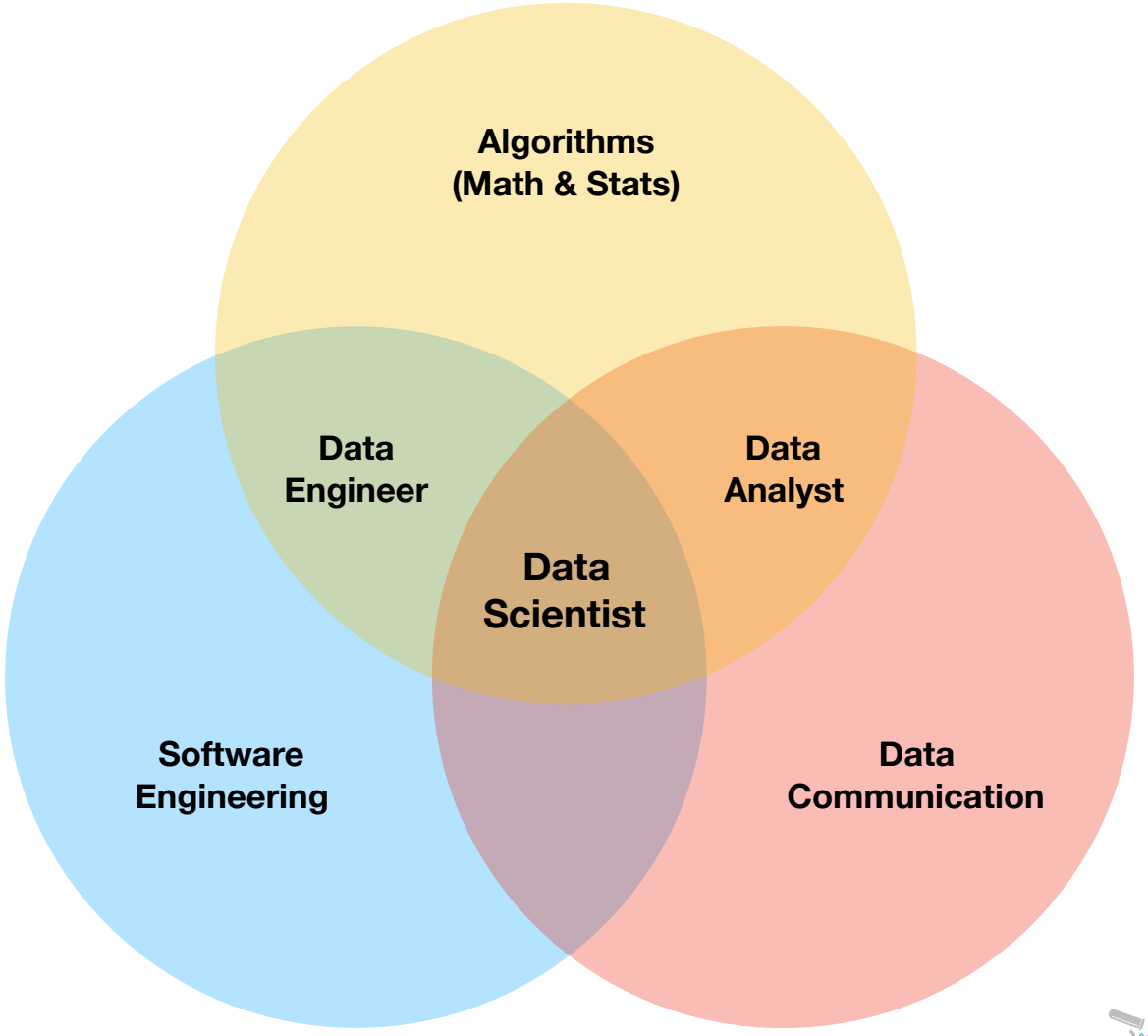
What is data science?

What is the goal of data science?

The Data Science Process



Data Scientist Skillset

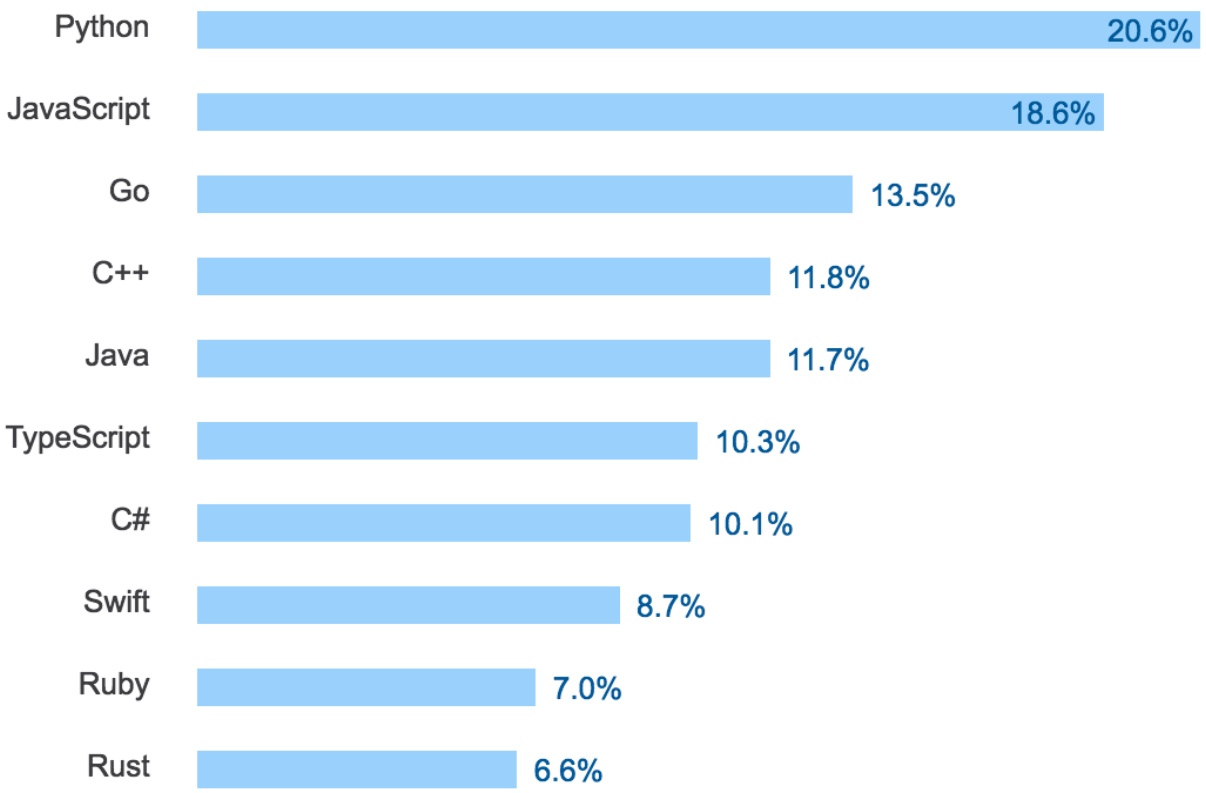


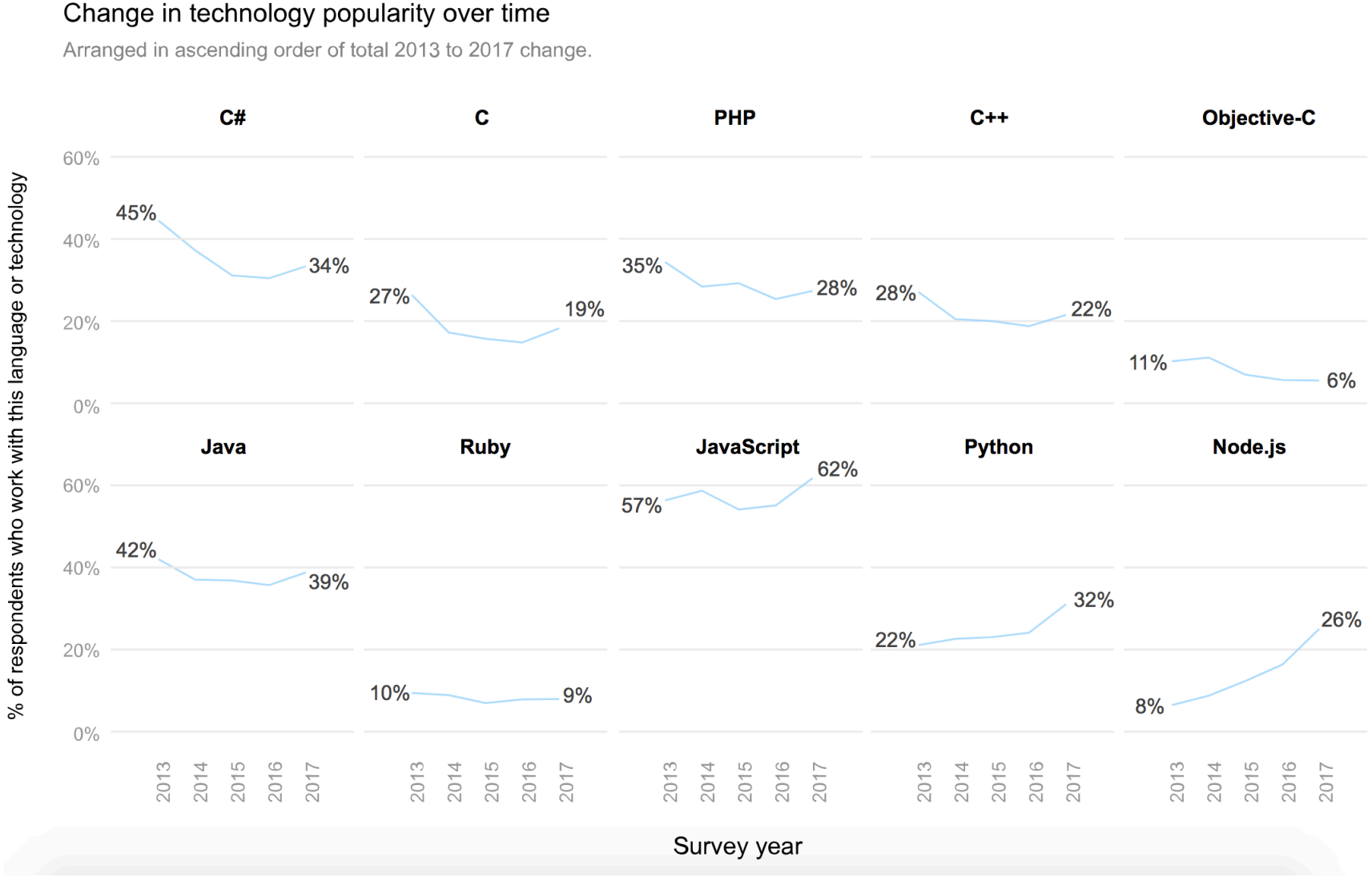
Software Engineering

- Build systems that can gather, process and store data
- Know 1+ programming languages
- Can apply your knowledge and skills to large datasets
- Familiar with distributed computing & the technologies

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted




Change in technology popularity over time
Arranged in ascending order of total 2013 to 2017 change.
C#
C
PHP
C++
Objective-C
Java
Ruby
JavaScript
Python
Node.js
45%
34%
27%
19%
35%
28%
28%
22%
11%
6%
42%
39%
10%
9%
57%
62%
22%
32%
8%
26%
60%
40%
20%
0%
2013
2014
2015
2016
2017
% of respondents who work with this language or technology
Survey year

Distributed Computing

- Linux
- Amazon Web Services (EC2, S3, RDS, etc.)
- Docker, Amazon ECS, Zookeeper, Kubernetes
- Apache Kafka
- Apache Spark, Apache Storm
- Redis, PostgreSQL, Cassandra
- HDFS

Suggested Engineering Path

- Python
- Amazon EC2 & S3
- PostgreSQL
- Apache Spark

Algorithms (Math & Stats)

- Understand stats, math and modeling rules
- Understand existing algorithms and create new ones
- Can design and measure experiments

Is ML Stats?

- Goal of both: learn from data.
- They are nearly the same and are converging.
- ML produces statistical models
- A model is a mathematical function defining the relationship between a set of inputs and a set of outputs.

Base Concepts to Learn

- Classification
- Regression
- Clustering
- Dimensionality reduction
- Model selection
- Preprocessing

Suggested Algorithms Path

- Jupyter Notebook
- pandas
- scikit-learn: classification, regression, clustering
- Spark MLlib: data science at scale

Data Communication

- Communicate the solution verbally.
- Communicate the solution with charty goodness.
- Relate the solution to business problems.
- Recommend actions to be taken using the solution.

Suggested Data Communication Path

- Jupyter Notebook
- Pandas
- Matplotlib
- Meetups & brown bag lunches

Ultimately it’s all about the users…

Two Objectives Orient Data Science to Users

Data Products

Improve product performance via automatic decision making.

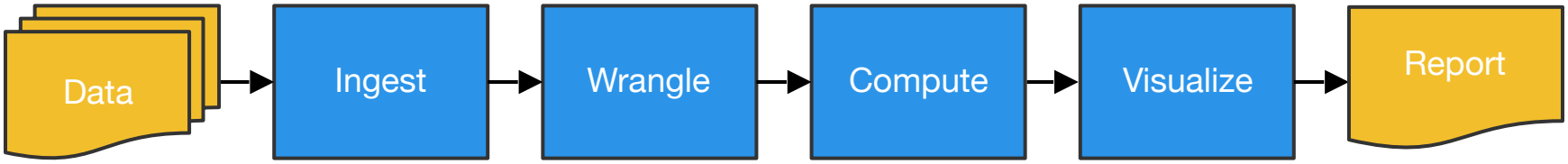
Decision Science

Inform strategy and key decisions by analysis of business metrics.

Data Products are Applications that

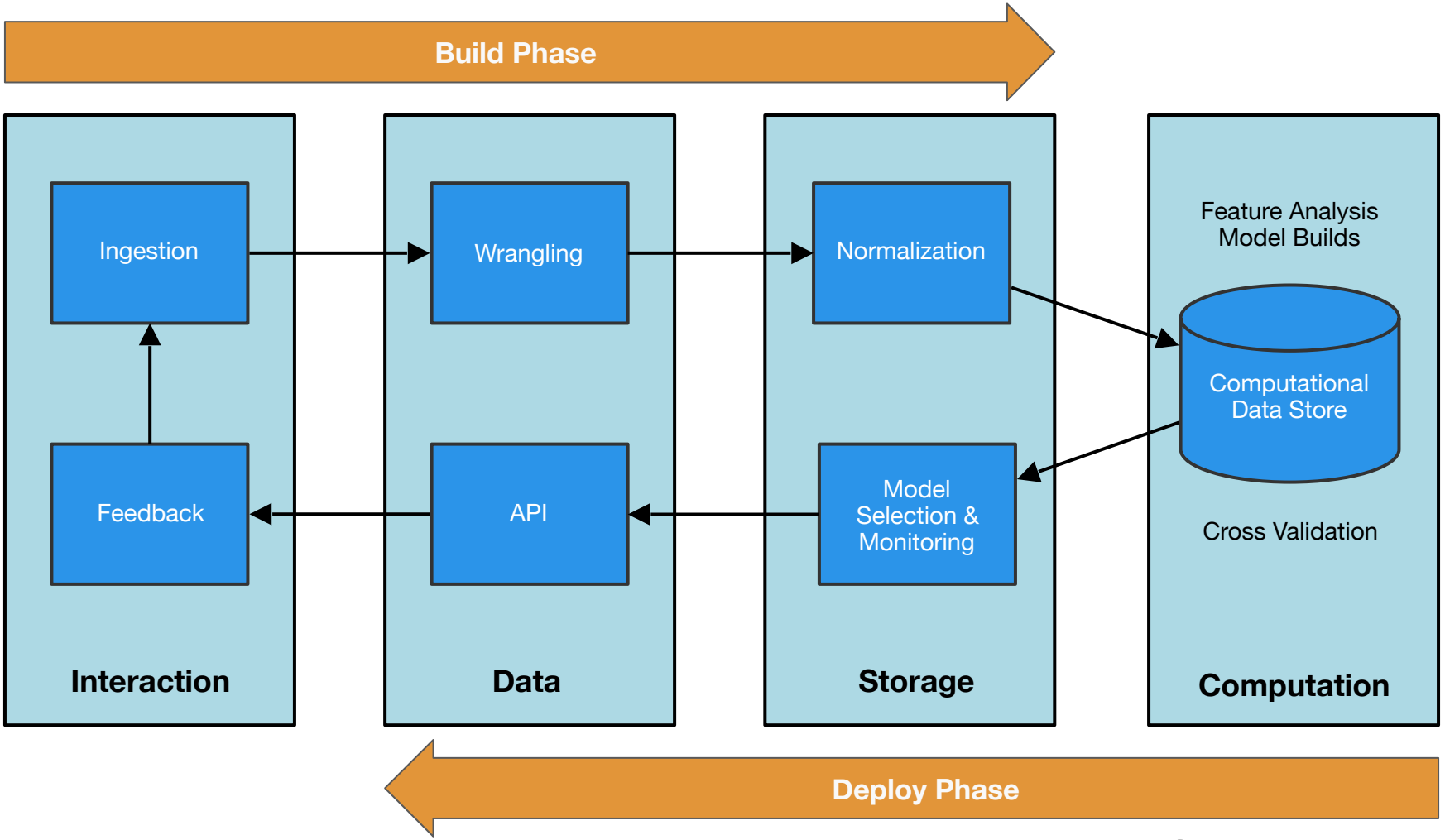
Employ Many Machine Learning Models

How Many Are Taught Data Science

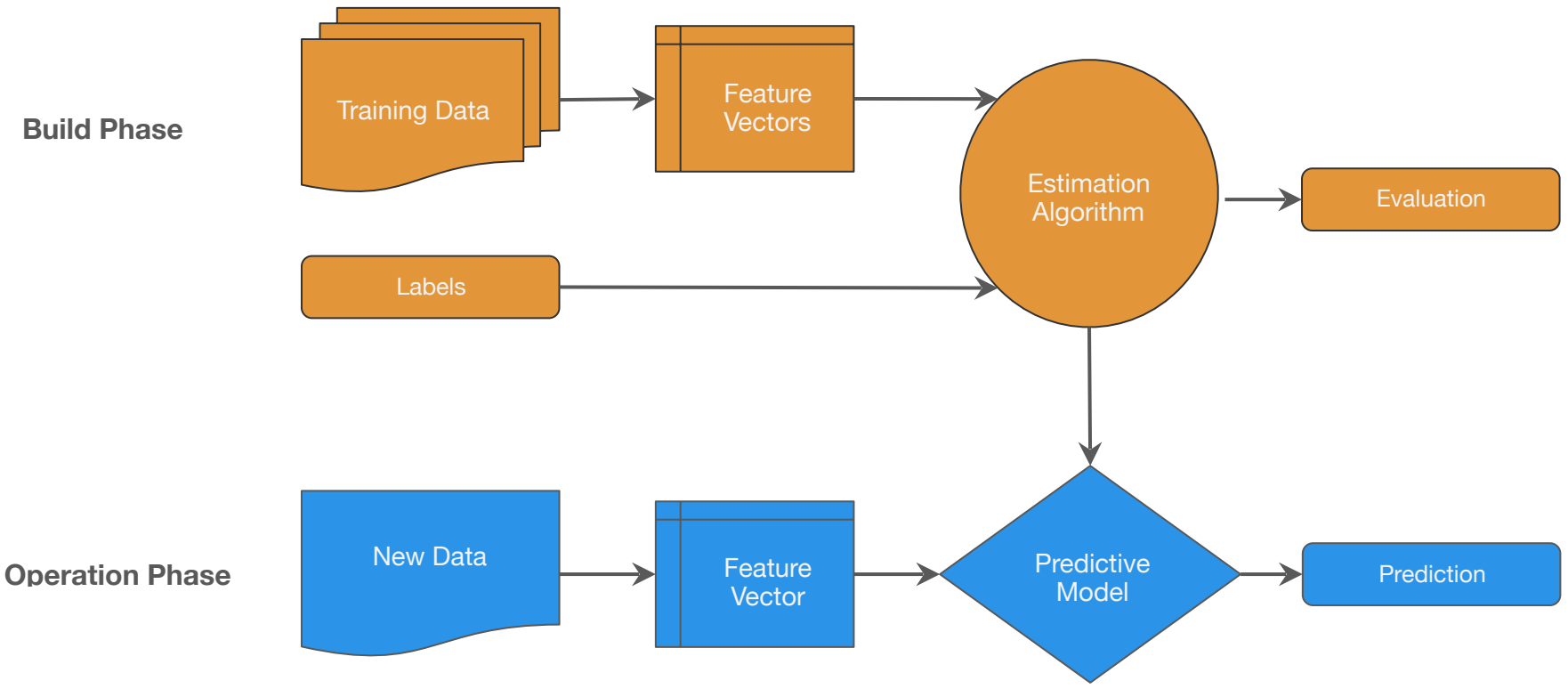



Without Feedback
Models are Disconnected
They cannot adapt, tune, or react.

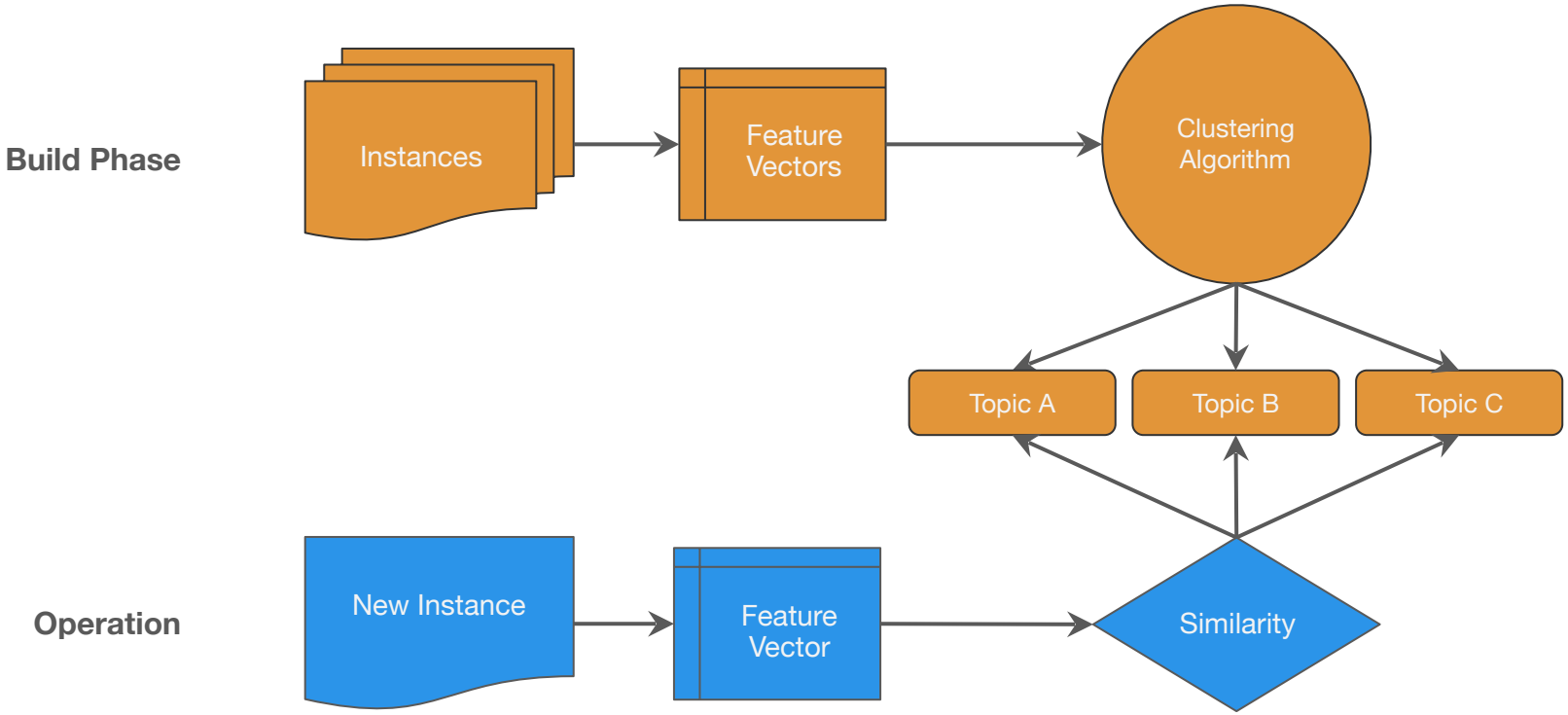
The Data Product Pipeline



Supervised Learning

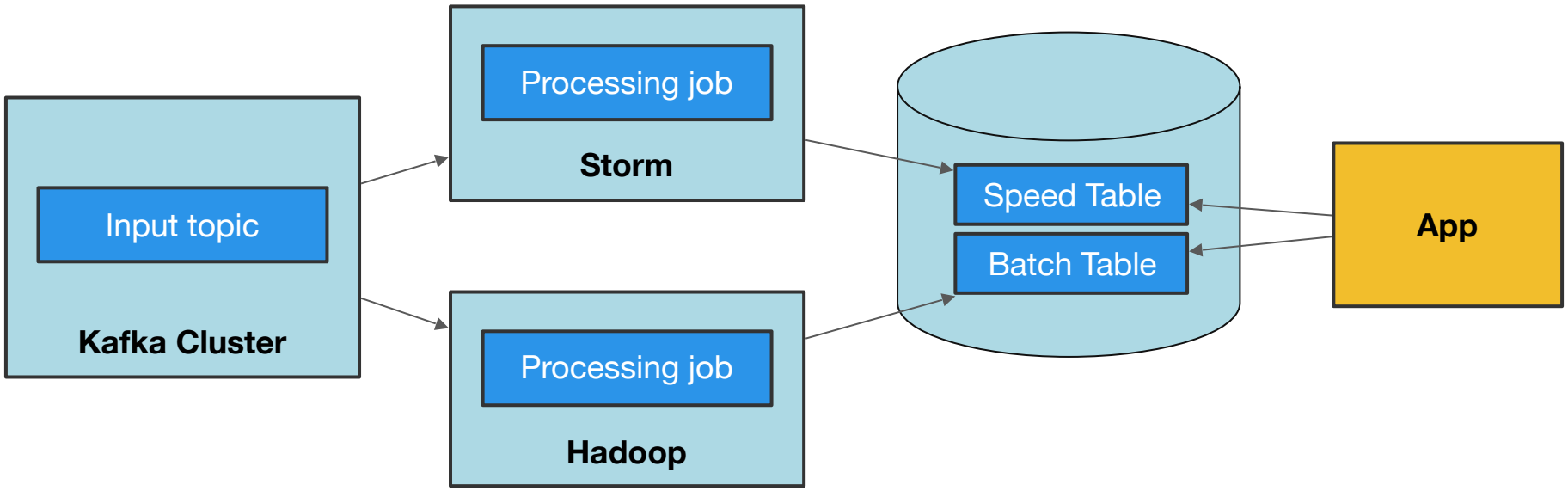


Unsupervised Learning

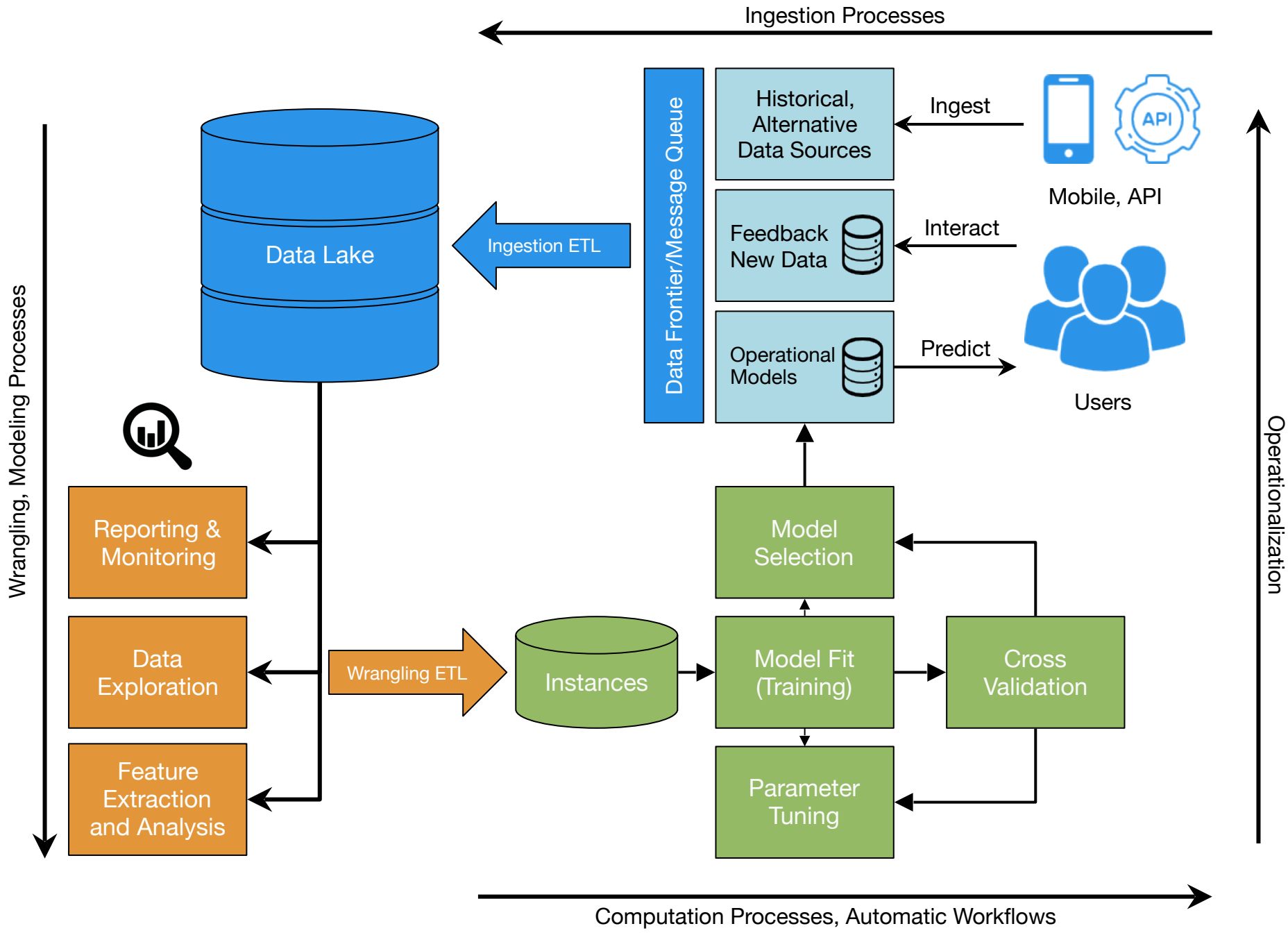



Data products aren’t single models
So how do we *architect* data products?

The Lambda Architecture



Questioning the Lambda Architecture -- Jay Kreps
http://oreil.ly/29RK7dQ


Ingestion Processes
Historical, Alternative Data Sources
Ingest
API
Mobile, API
Data Frontier/Message Queue
Feedback New Data
Interact
Operational Models
Predict
Users
Data Lake
Ingestion ETL
Wrangling, Modeling Processes
Operationalization
Reporting & Monitoring
Data Exploration
Feature Extraction and Analysis
Wrangling ETL
Instances
Model Selection
Model Fit (Training)
Cross Validation
Parameter Tuning
Computation Processes, Automatic Workflows

Semantic Search

Enabling and improving search results for a domain specific application.

Recommendation

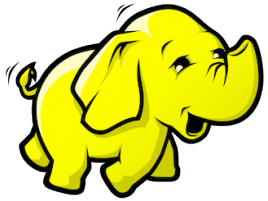
Creating collective models using collaborative filtering and graphs.

Political Classifier

Create per-user models as well as global models for dynamic interaction.

Semantic Search

Analyst Architecture



Topic Modeling

Sampling

Entity Extraction

Ranking

Web Scraping

Preprocessing

Frame Modeling

Search Service

News Articles
Reports
Images
Videos

Data Lake

Model Storage

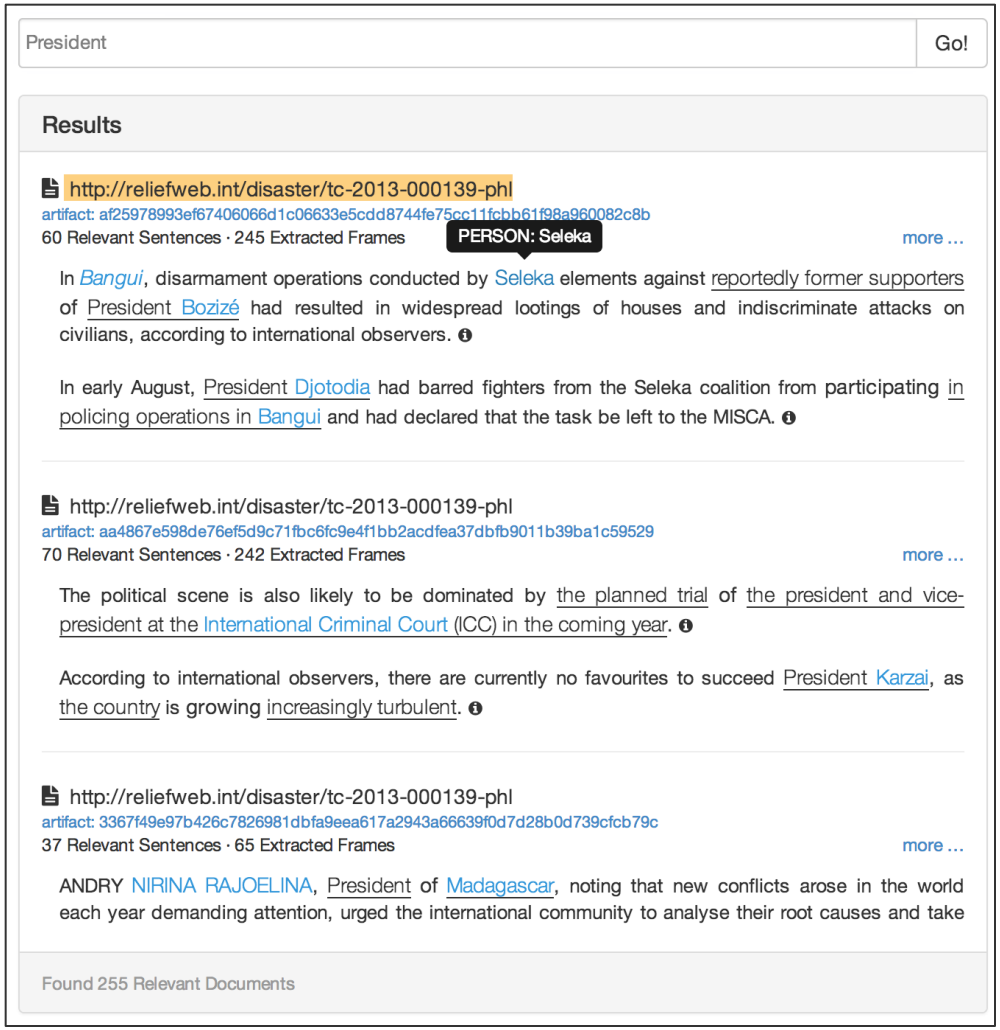
Annotated Data Store

Semantic Search

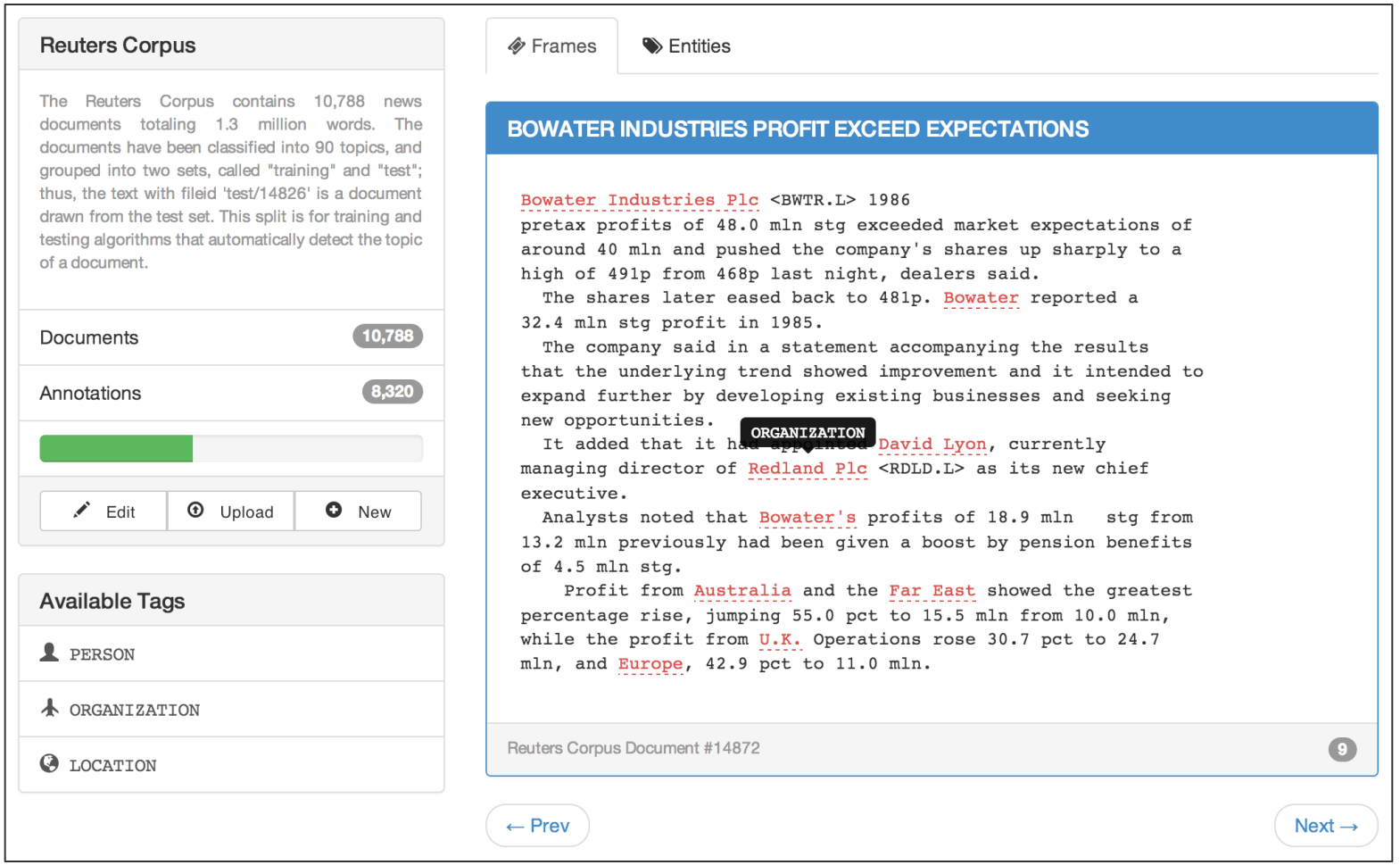
Model Evaluation and Interaction

Annotating & Report Generation

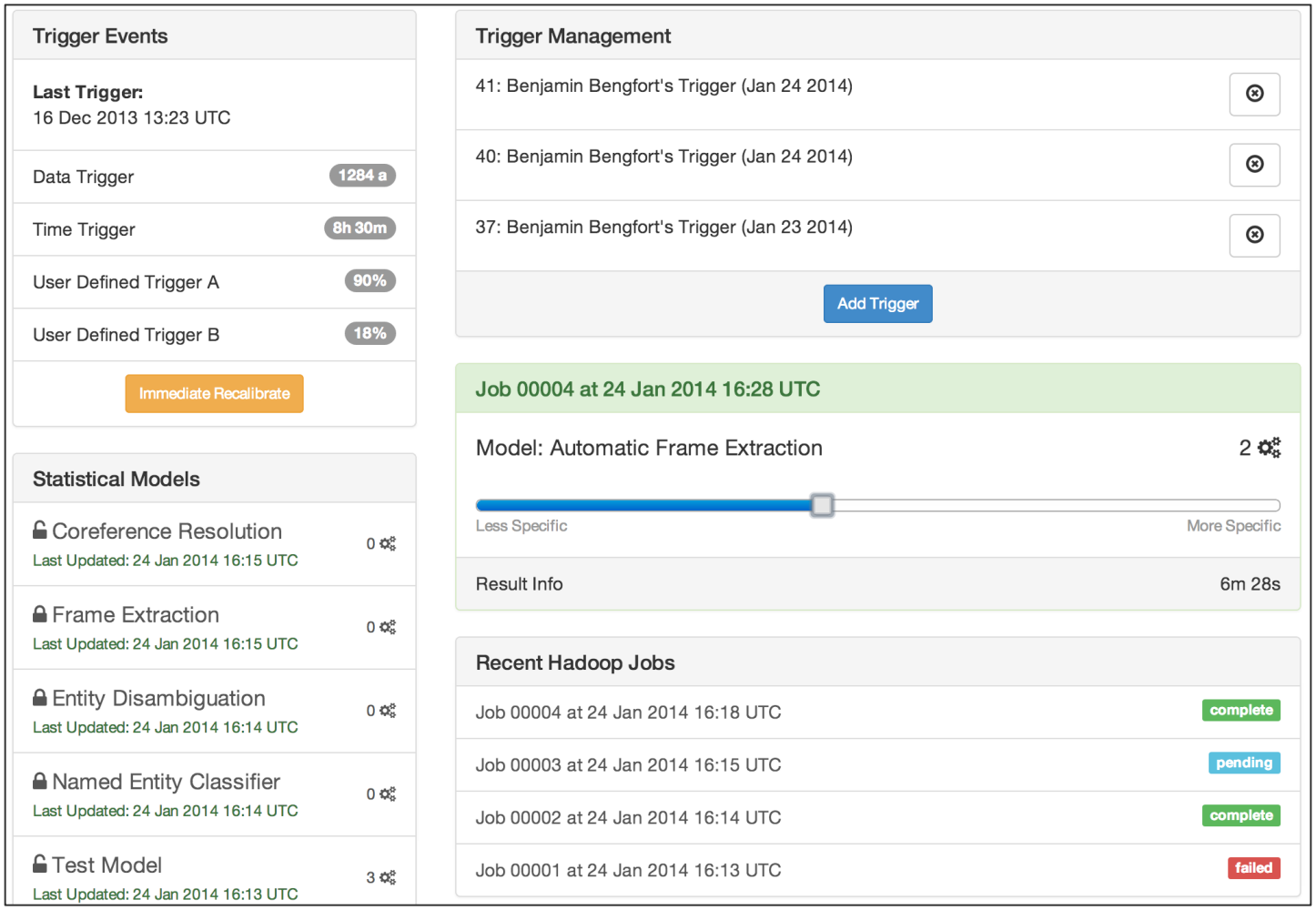
Analyst Architecture: Search



Analyst Architecture: Document Revie

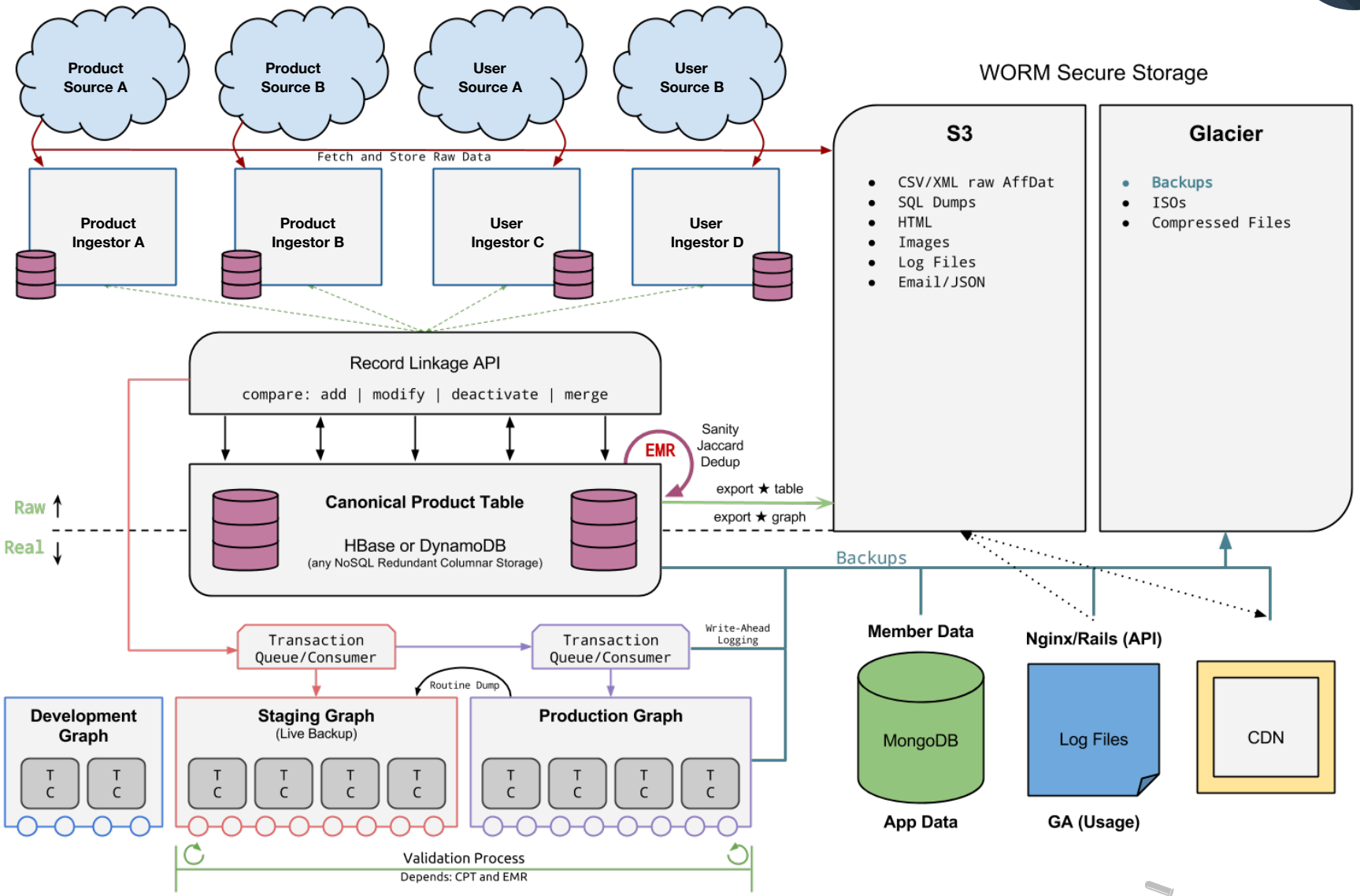


Analyst Architecture: Triggers

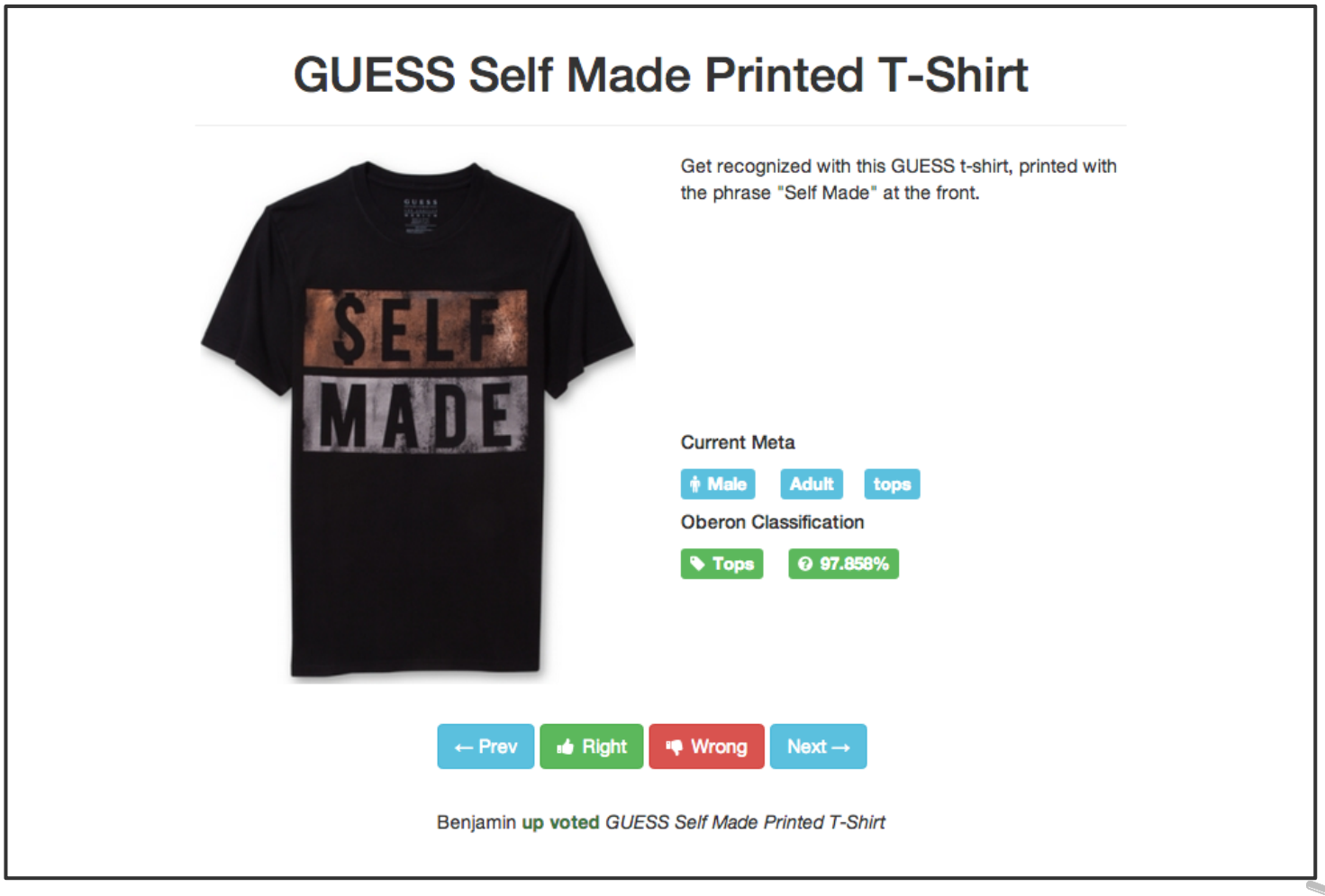


Recommendation

Recommender Architecture

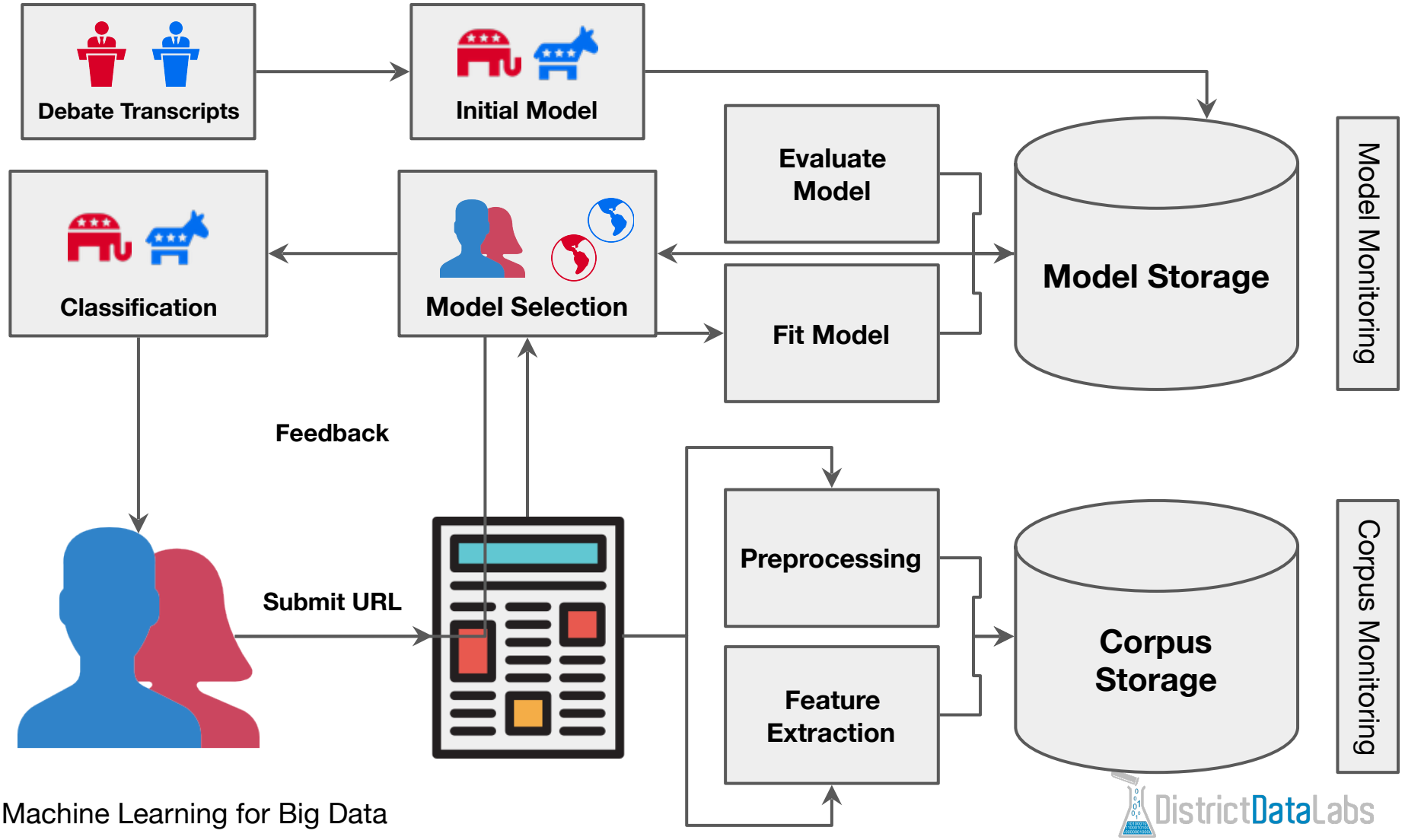


Recommender: Annotation Service

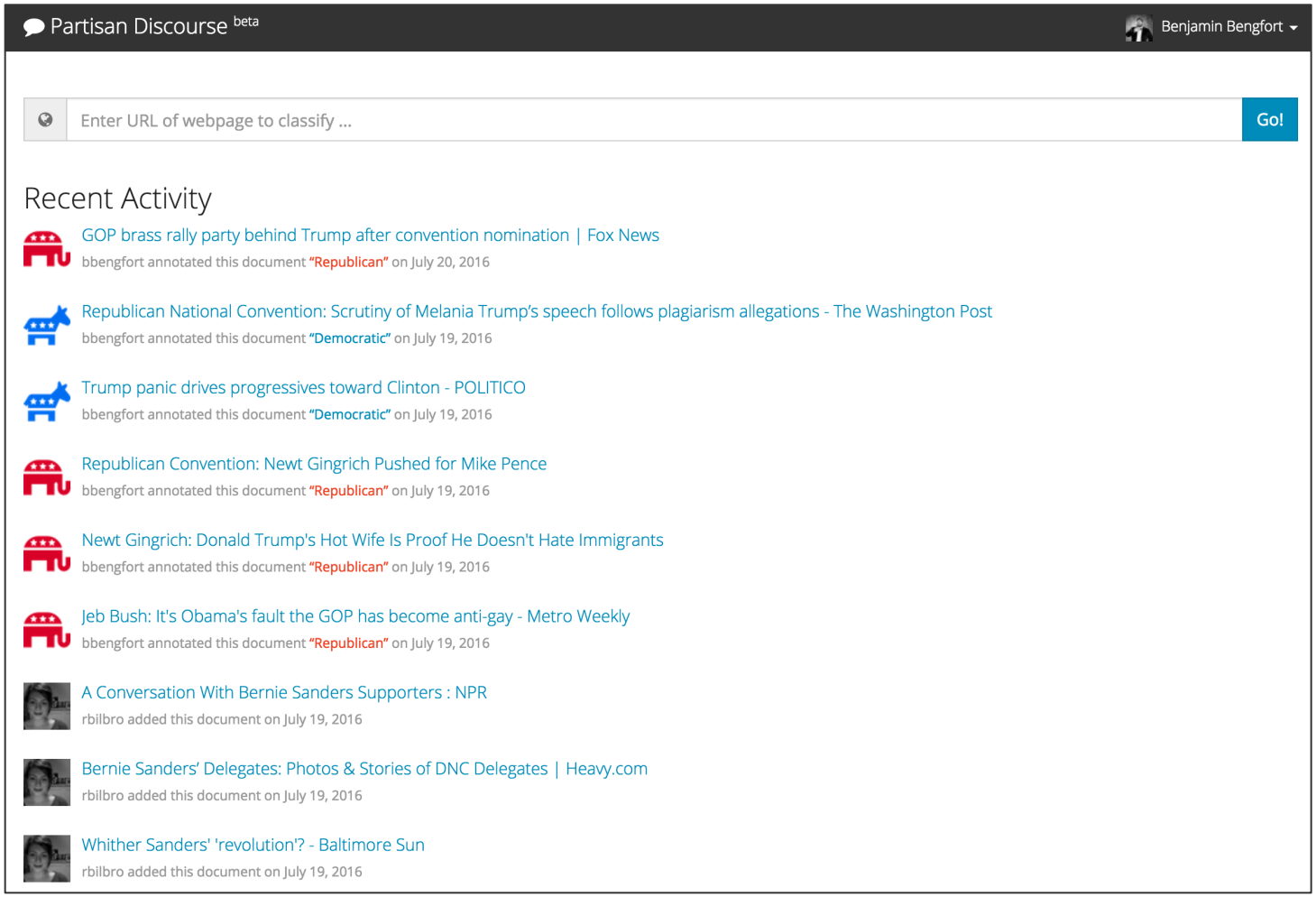


Political Classifier

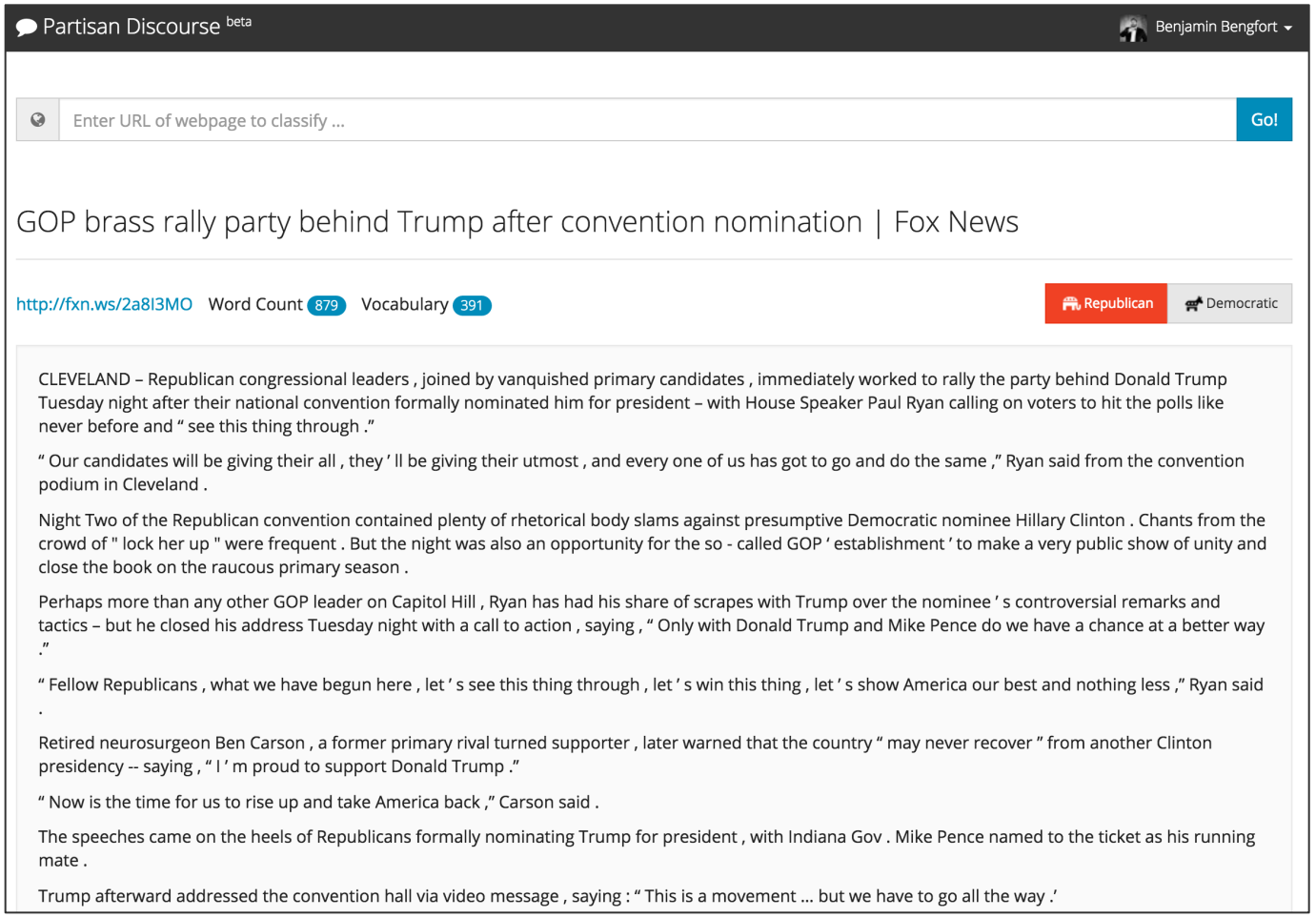
Partisan Discourse Architecture



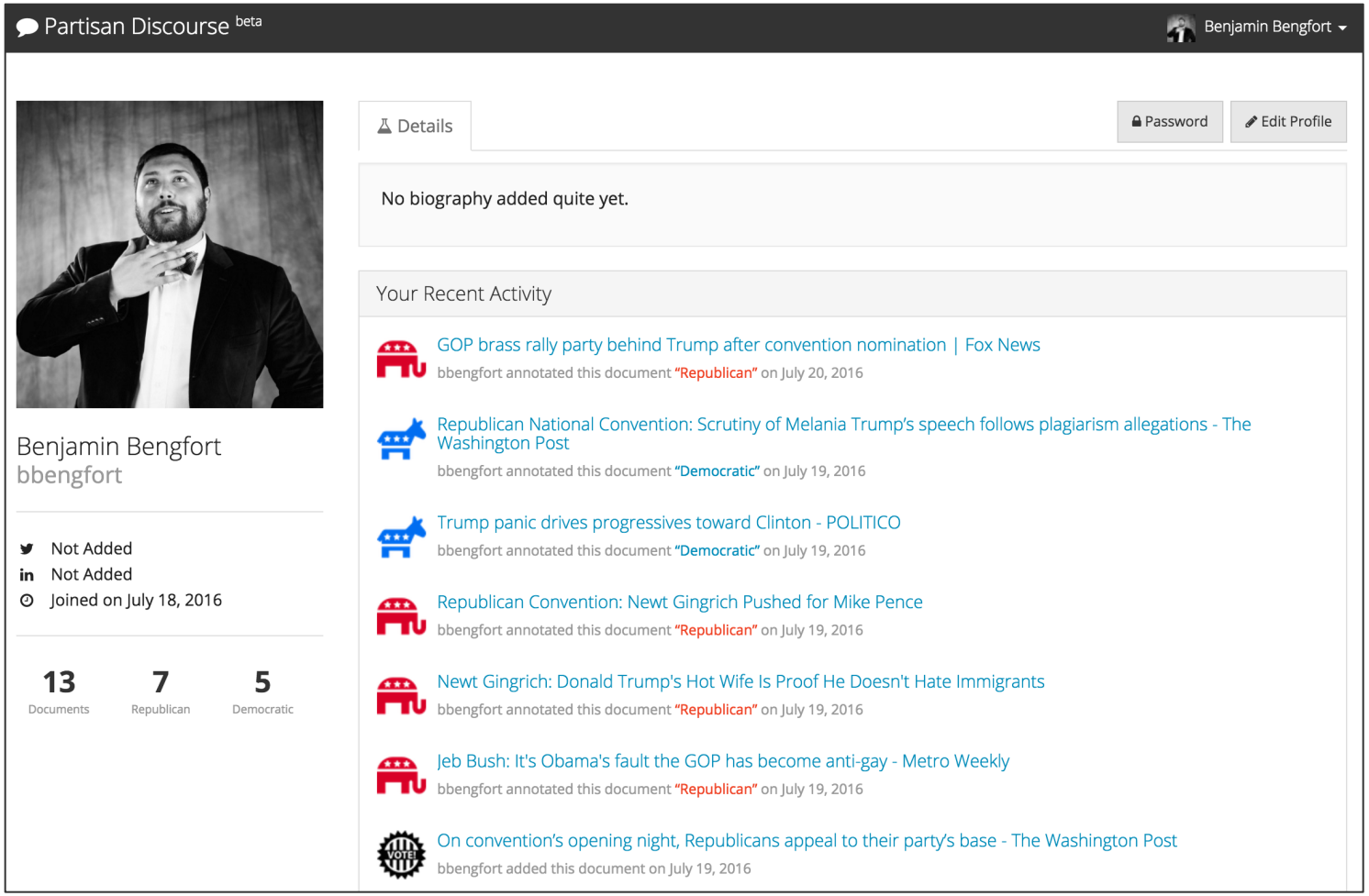
Partisan Discourse: Adding Document



Partisan Discourse: Documents



Partisan Discourse: User Specific Mod



Key Questions for Data Product Architectures

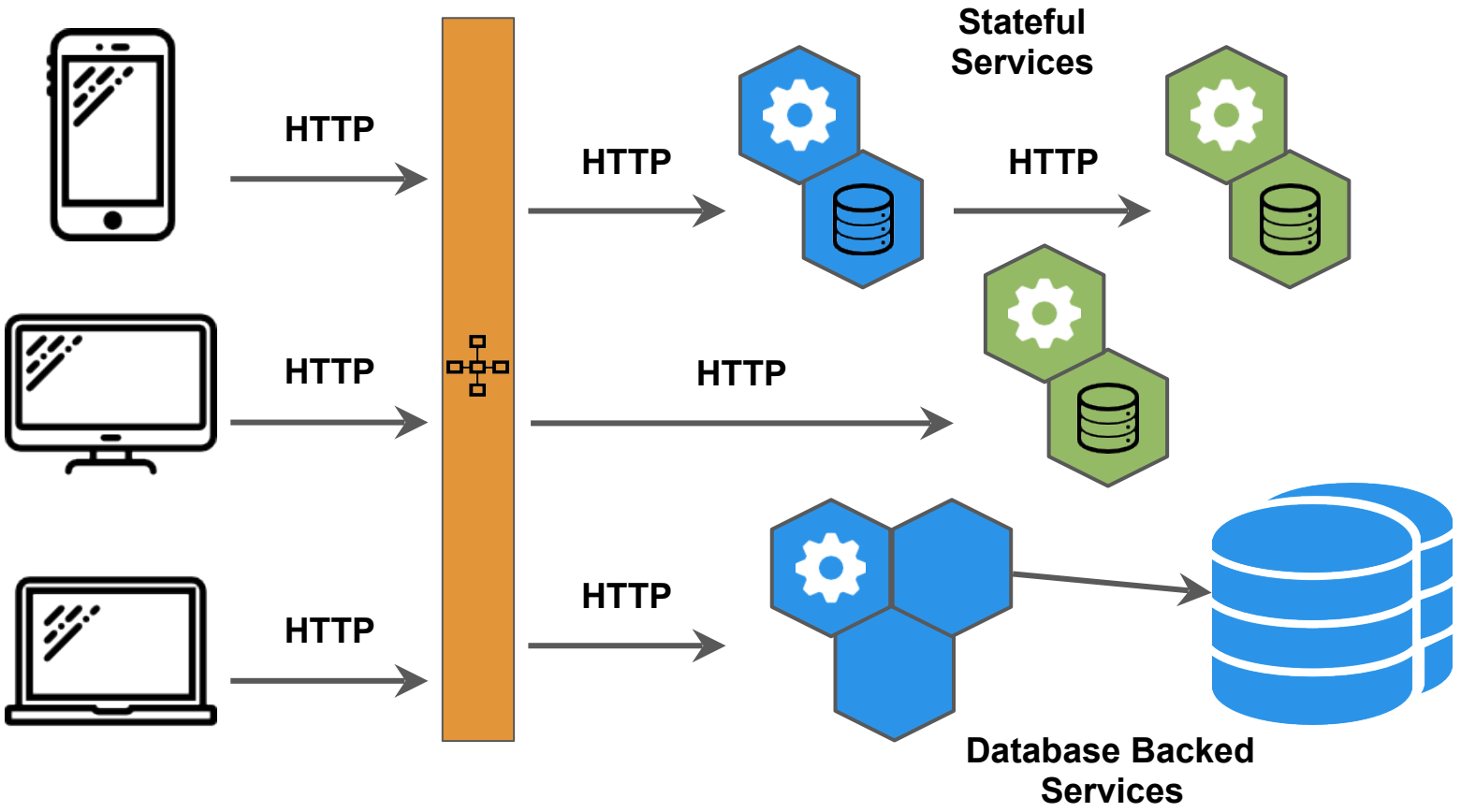
How do you detect deterioration?

When do you rebuild (triggers)?

How to monitor change and decay old data?

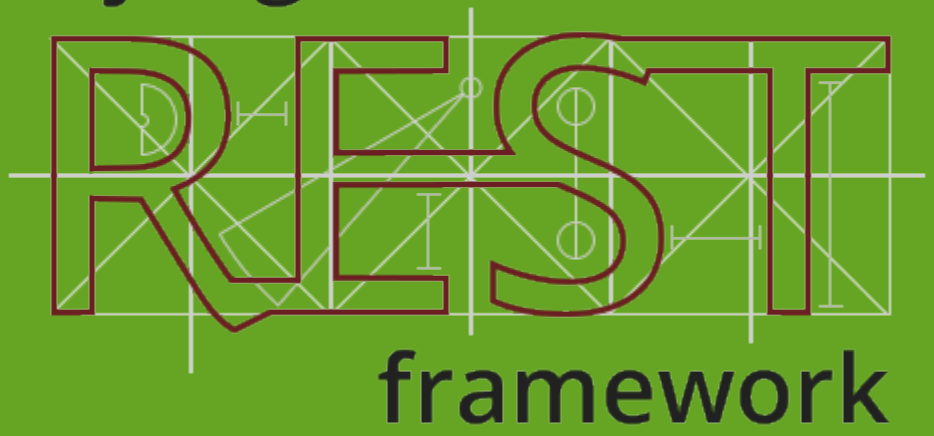
How do you ensemble models to improve?

Microservices Architecture



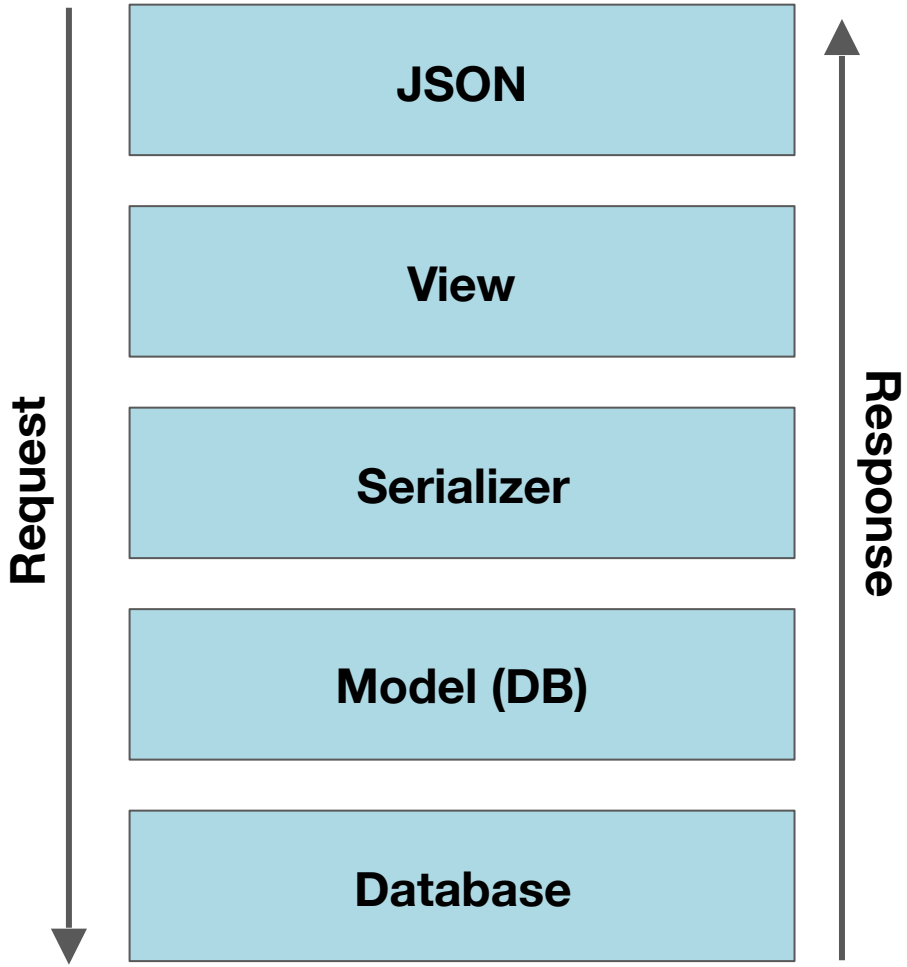
Martin Fowler http://bit.ly/2a2Lc17

django

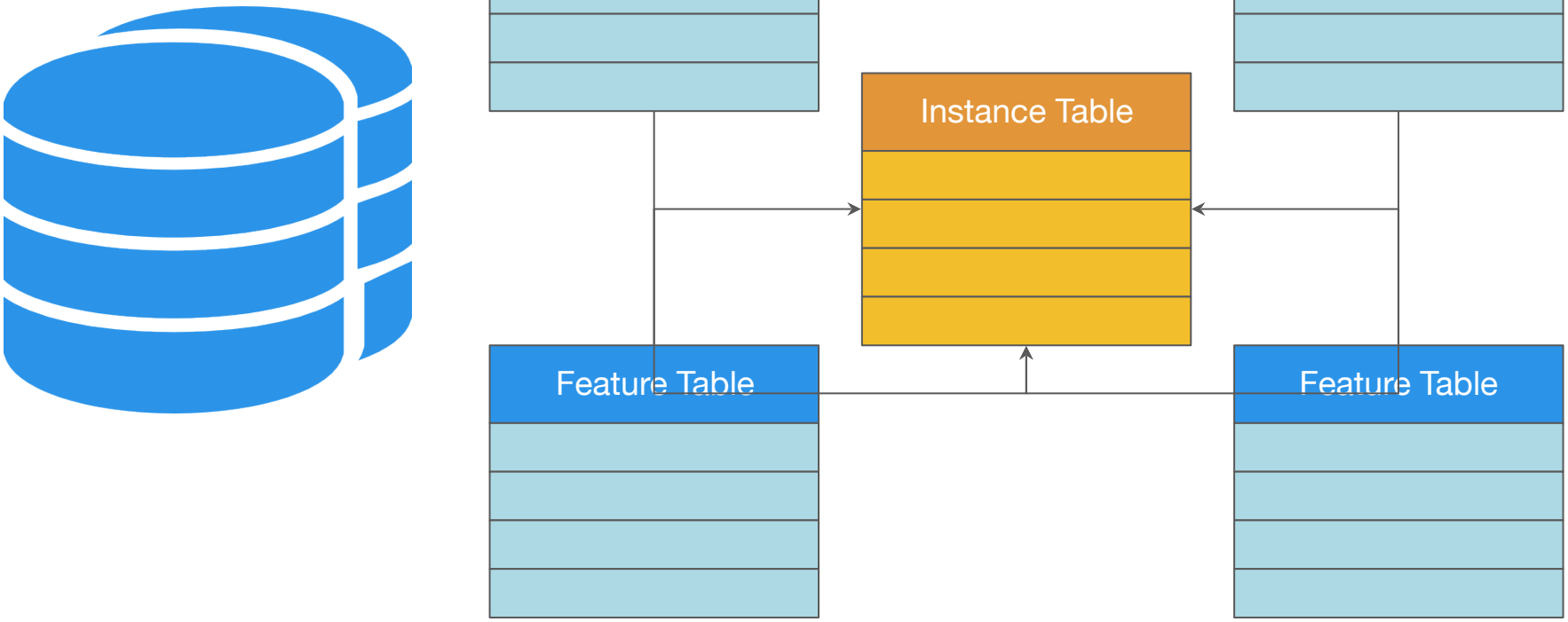


Django Application Model

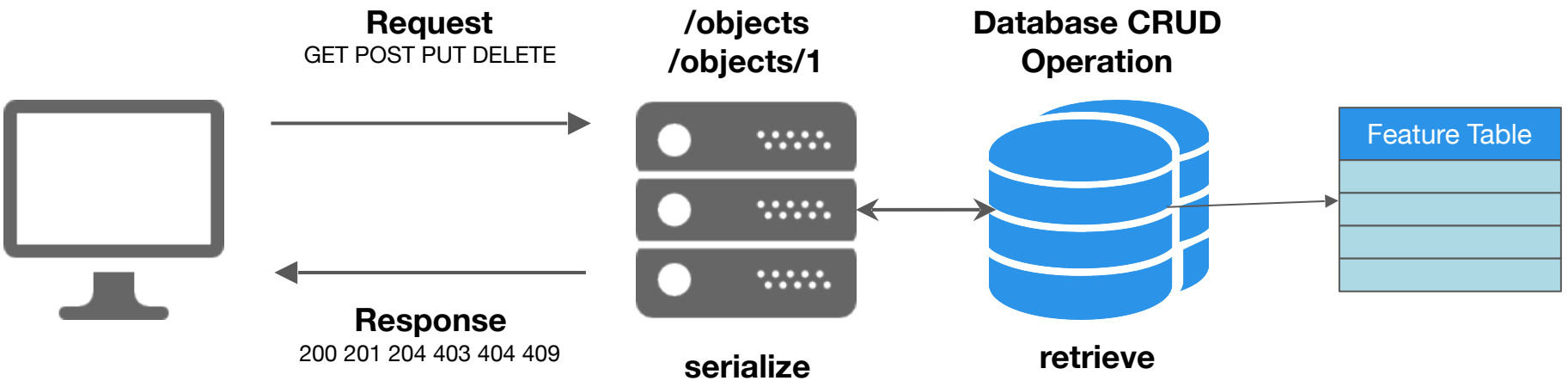
- Input and output of the application is JSON.
- Views are associated with endpoints and handle requests and responses.
- Serializers deal with the JSON database
- Models interact with the database.
- Almost any database backend is supported.



Features and Instances as Star Schema



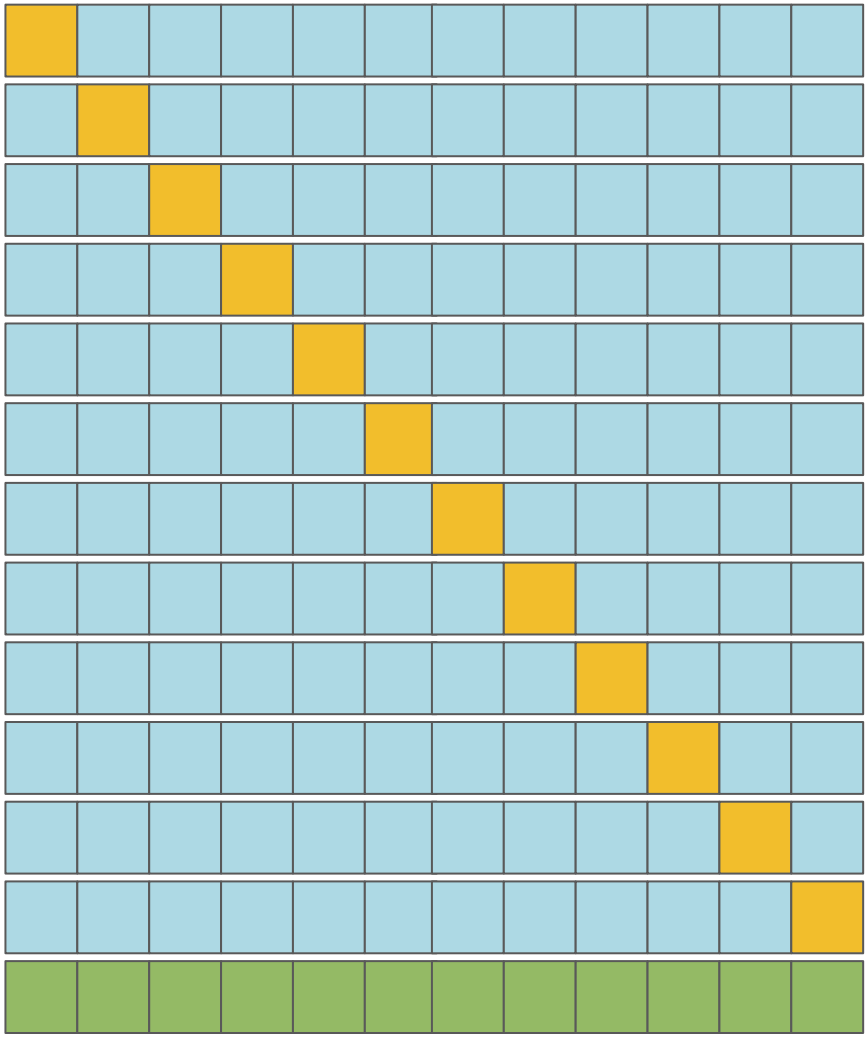
REST API Feature Interaction



Model (ML) Build Process: Export Instance Table

```
COPY (
    SELECT instances.* FROM instances
        JOIN feature on feature.id =
instance.id
        ...
    ORDER BY instance.created DESC
    LIMIT 10000
) as instances
TO '/tmp/instances.csv' DELIMITER ',' CSV
HEADER;
```

Model (ML) Build Process: Build Model



```
import pandas as pd
from sklearn.svm import SVC
from sklearn.cross_validation import
KFold

# Load Data
data    = pd.read_csv('/tmp/
instances.csv')
scores = []

# Evaluation
folds = KFold(n=len(data), n_folds=12)
for train, test in folds:
    model = SVC()
    model.fit(data[train])
    score = model.score(data[test])
    scores.append(score)

# Build the actual model
model = SVC()
model.fit(data)
```

Model (ML) Build Process: Store Model

```python
import json
import pickle
import base64
import datetime

data = pickle.dump(model)
data = base64.base64encode(data)

return {
    "model": data,
    "created": datetime.datetime.now(),
    "form": repr(model),
    "name": model.__class__.__name__,
    "scores": scores,
}
```

Model Data Storage

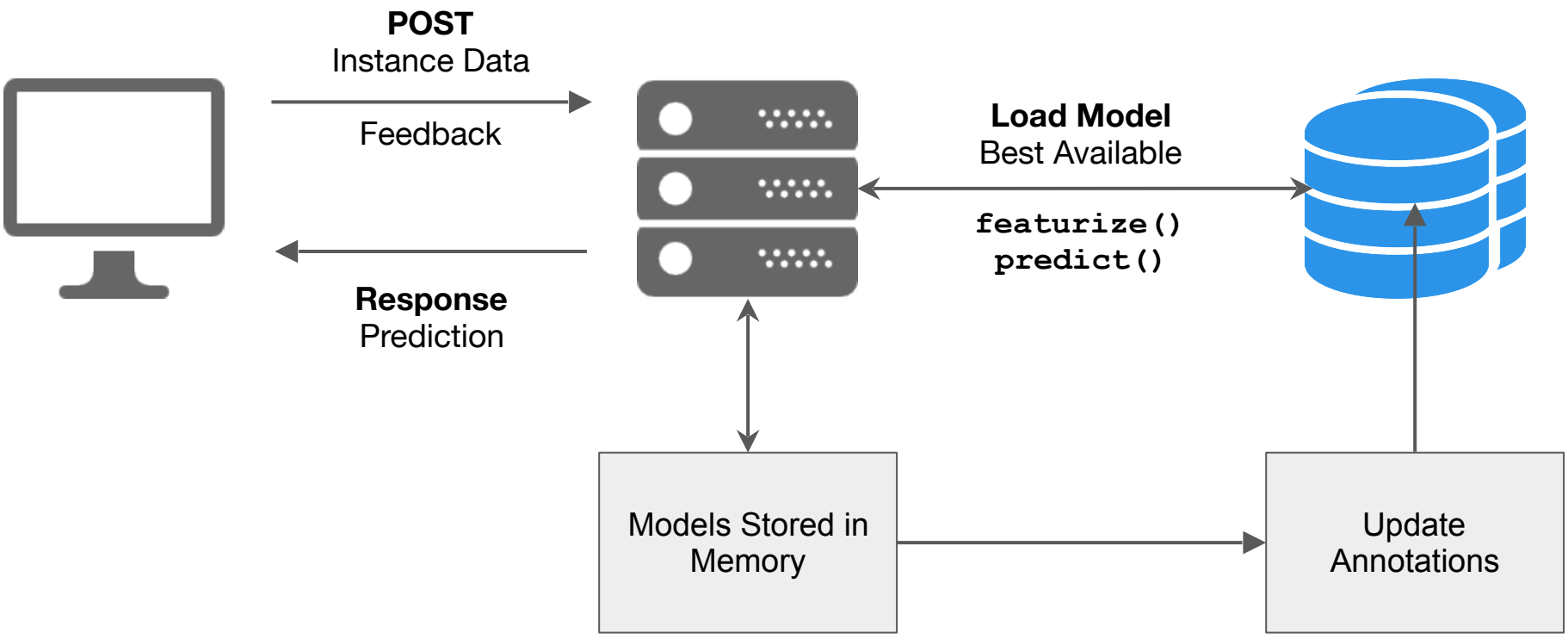
```
from django.db import models

class PredictiveModel(models.Model):

    name      = models.CharField()
    params    = models.JSONField()
    build     = models.FloatField()
    f1_score  = models.FloatField()
    created   = models.DateTimeField()
    data      = models.BinaryField()
```

Considerations

- FileField vs. BinaryField
- How much meta data can you store (the more the better)
- What fields can be monitored in an model selection management system? (E.g. CMS for ML))

ID	Model	Hyperparameters	Build Time	F1	Pickle
1	Naive Bayes	{“alpha”: 1.0}	235.32	.832	BLOB
2	SVC	{“C”: 1.0, “kernel”: “linear”}	20.312	.861	BLOB
3	KNN	{“k”: 5, “weights”: “distance”}	482.129	.821	BLOB

REST API Model Interaction



Hands-On Lab

Tasks

Log in to the server.

Create a folder with your personal ID

I'll copy the code into the folders for you.

Let's view and run the notebooks:

1. Easy as Pi
2. Load a CSV File
3. Word Count

Hadoop: An Operating System for Big Data

Hadoop Architecture

Distributed System Requirements

Fault Tolerance - If a component fails, it should not result in the failure of the entire system.

Recoverability - In the event of a failure, no data should be lost.

Consistency - The failure of one job or task should not affect the final result.

Scalability - Adding load leads to a decline in performance, not failure

Hadoop Architecture - Basic Concepts

Data distributed immediately and stored on multiple nodes.

Stored in blocks of fixed size (128MB) and each block duplicated multiple times.

Computation referred to as a job, and jobs broken up into tasks.

Jobs are written at high level without concern for network programming, time, or low-level infrastructure.

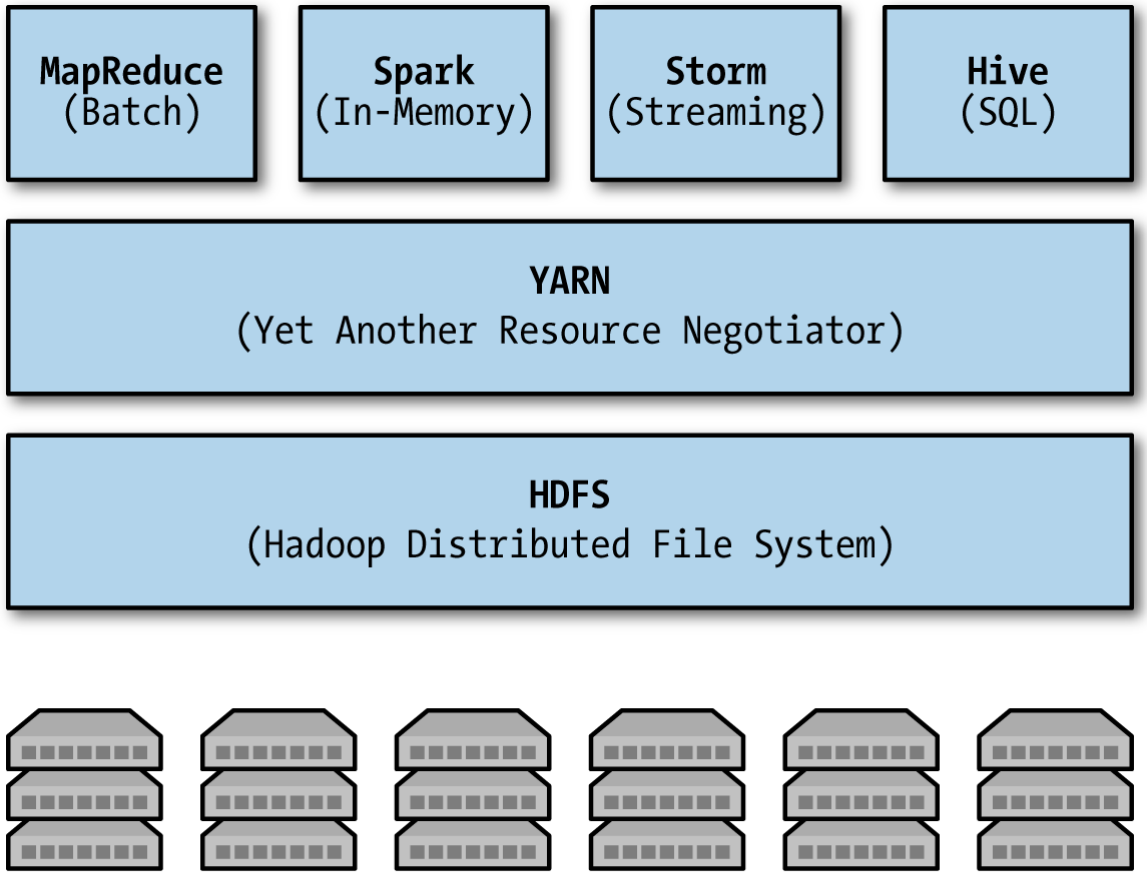
Hadoop Architecture - Basic Concepts

Amount of network traffic should be minimized transparently by the system.

Jobs are fault tolerant, usually through task redundancy.

Master programs allocate work to worker nodes such that many worker nodes can operate in parallel.

Hadoop Architecture



What's In a Cluster?

What is a Cluster?

Set of machines running HDFS and YARN.

Individual horizontally scalable machines called **nodes**.

HDFS and YARN implemented by **daemon** processes.

Daemons run inside Java Virtual Machine (JVM).

Each mode identified by type of process it runs.

Types of Nodes

Master Nodes

Coordinate services for Hadoop workers.

Are usually the entry point for user access.

Worker Nodes

Most computers in the cluster.

Run services that accept tasks from Master Nodes.

Implement both HDFS and YARN worker services.

HDFS Services

NameNode (Master) - Stores directory tree, file metadata, and file locations.

Secondary NameNode (Master) - Performs housekeeping and checkpointing on behalf of NameNode.

DataNode (Worker) - Stores and manages HDFS blocks on the local disk and reports health and status to NameNode.

YARN Services

Resource Manager (Master) - Allocates and monitors available cluster resources and schedules jobs.

Application Manager (Master) - Coordinates applications being run on the cluster.

NodeManager (Worker) - Runs and manages processing tasks and reports health and status of tasks.

MapReduce vs. YARN

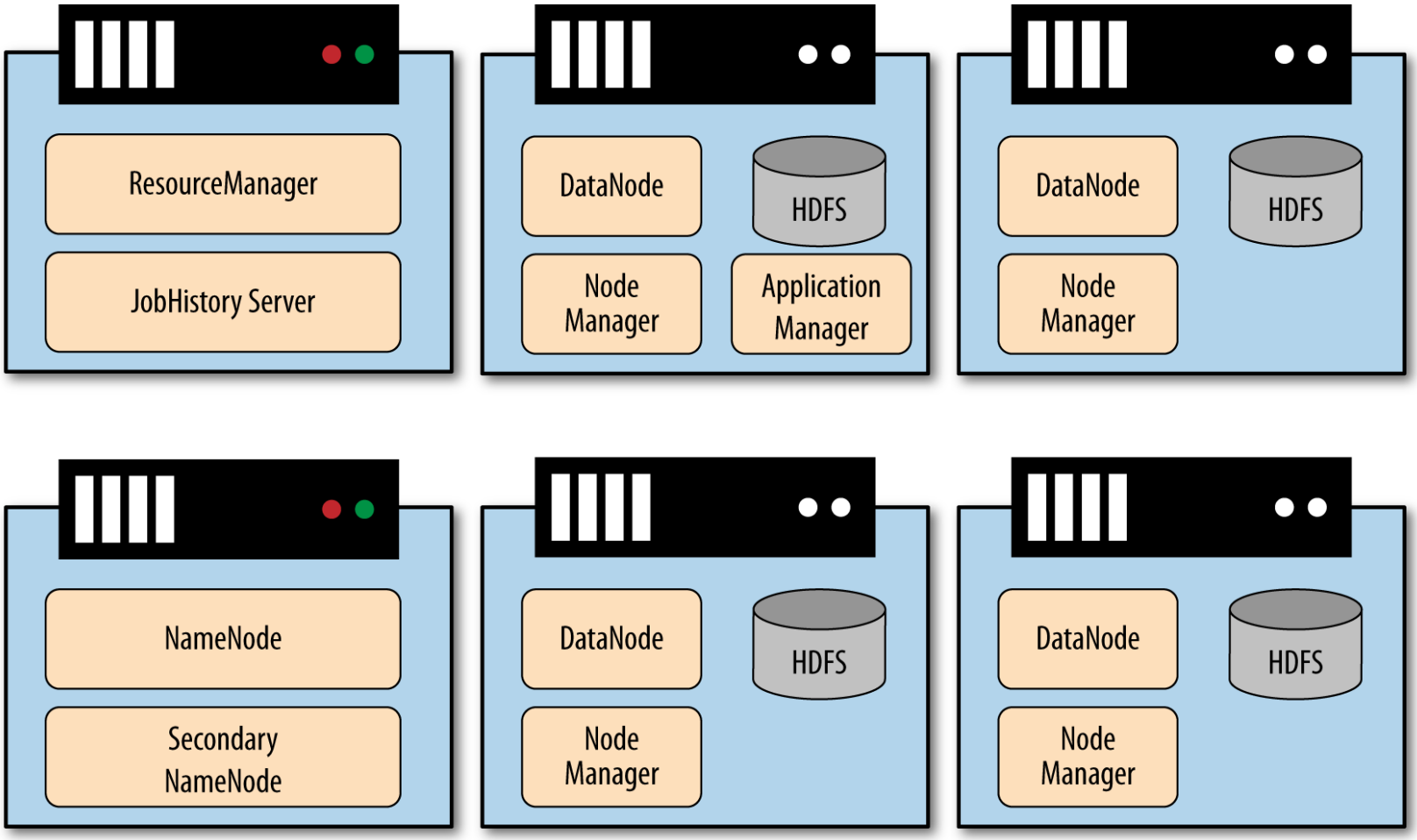
MapReduce

- Very efficient for large-scale batch workloads, but also I/O intensive.
- Limitations for interactive analysis, graph processing, machine learning, and other memory-intensive algorithms.

YARN

- Decouples workload management from resource management.
- Multiple applications can share centralized, common resource management service.

Cluster Example



HDFS Caveats

HDFS Caveats

Performs with modest number of very large files.

Implements the WORM (write once, read many) pattern.

Optimized for large, streaming file reads.

Best suited for storing raw input, intermediary results, and final results.

Not a good fit as a data backend for applications.

Blocks

HDFS files are split into blocks.

Typically 64MB, 128MB, or 256MB.

Allow large files to be distributed across many machines.

Replicated across DataNodes (3-fold by default)

Data Management

Master NameNode tracks
- What blocks make up a file.
- Where blocks are located.

Secondary NameNode merges snapshot of current data space with edit log.

When client application wants to read a file:
- Requests metadata from NameNode to locate blocks.
- Requests locations of DataNodes that store blocks.