

STITCH FIX

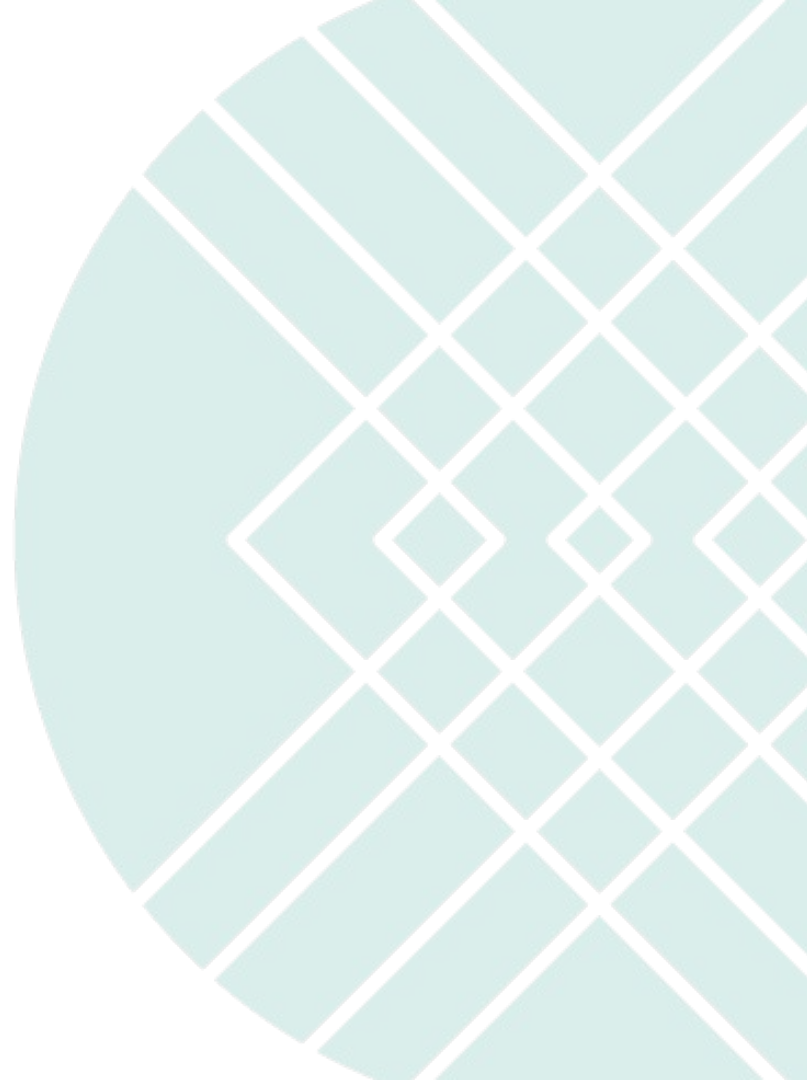
Data Science in the Cloud

Stefan Krawczyk

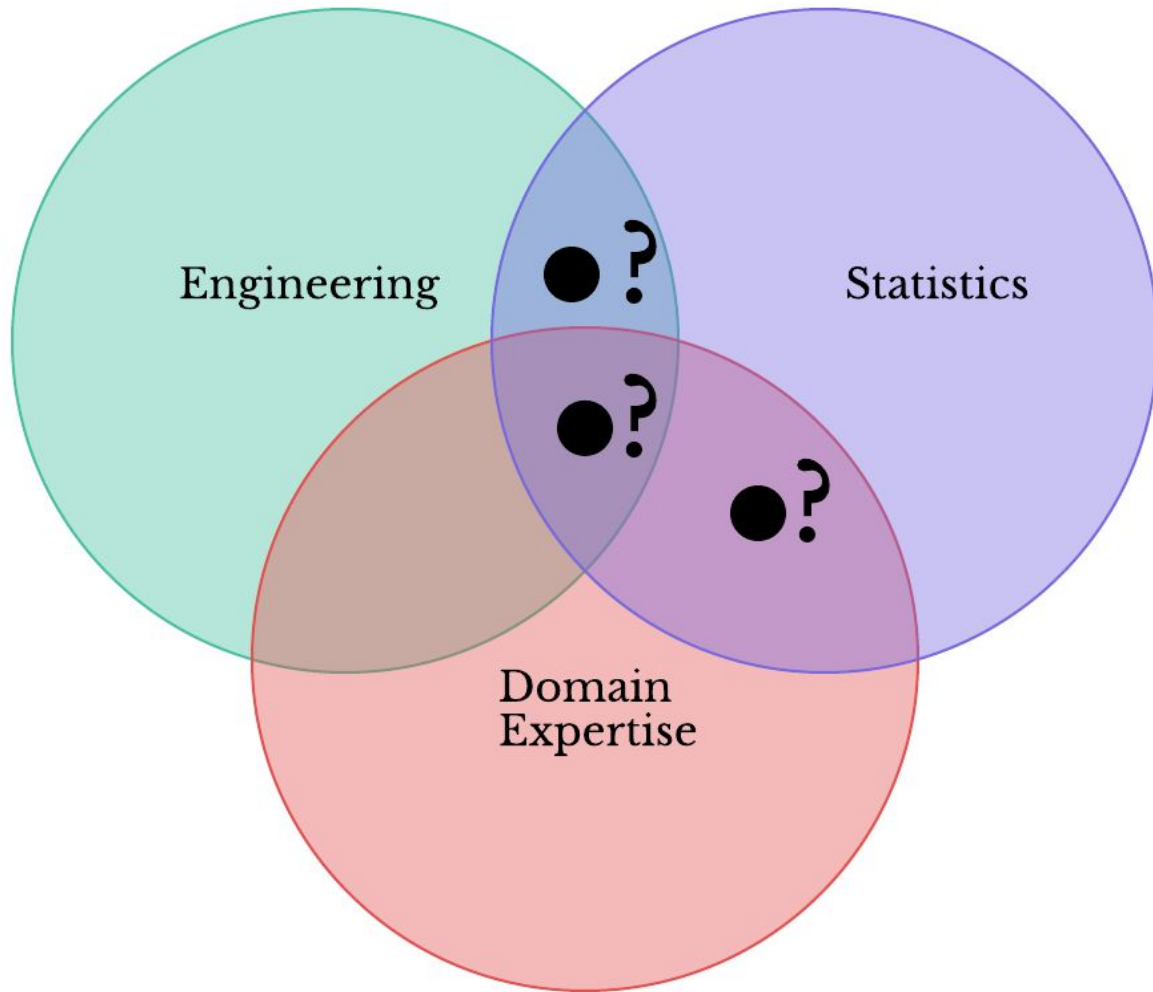
 @stefkrawczyk

 [linkedin.com/in/skrawczyk](https://www.linkedin.com/in/skrawczyk)

November 2016



Who are Data Scientists?





(((Josh Wills)))

@josh_wills

 [Follow](#)

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

9:55 AM - 3 May 2012



1,500



1,068



(((Josh Wills)))

@josh_wills



Follow

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

9:55 AM - 3 May 2012



1,500



1,068

Means: skills vary wildly

But they're in
demand and expensive

**“The Sexiest Job
of the 21st Century”
- HBR**

How many
Data Scientists do you have?

At Stitch Fix we have ~80

~85% have not done formal CS

But what do they do?

What is Stitch Fix?

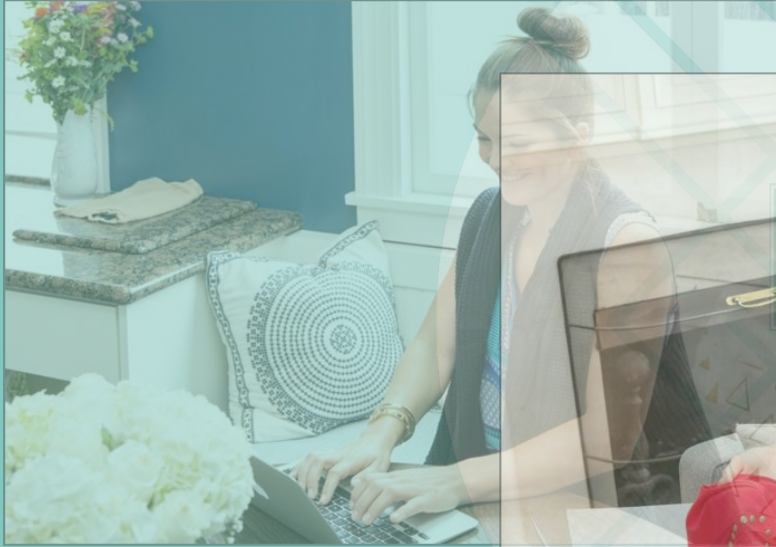
STITCH FIX



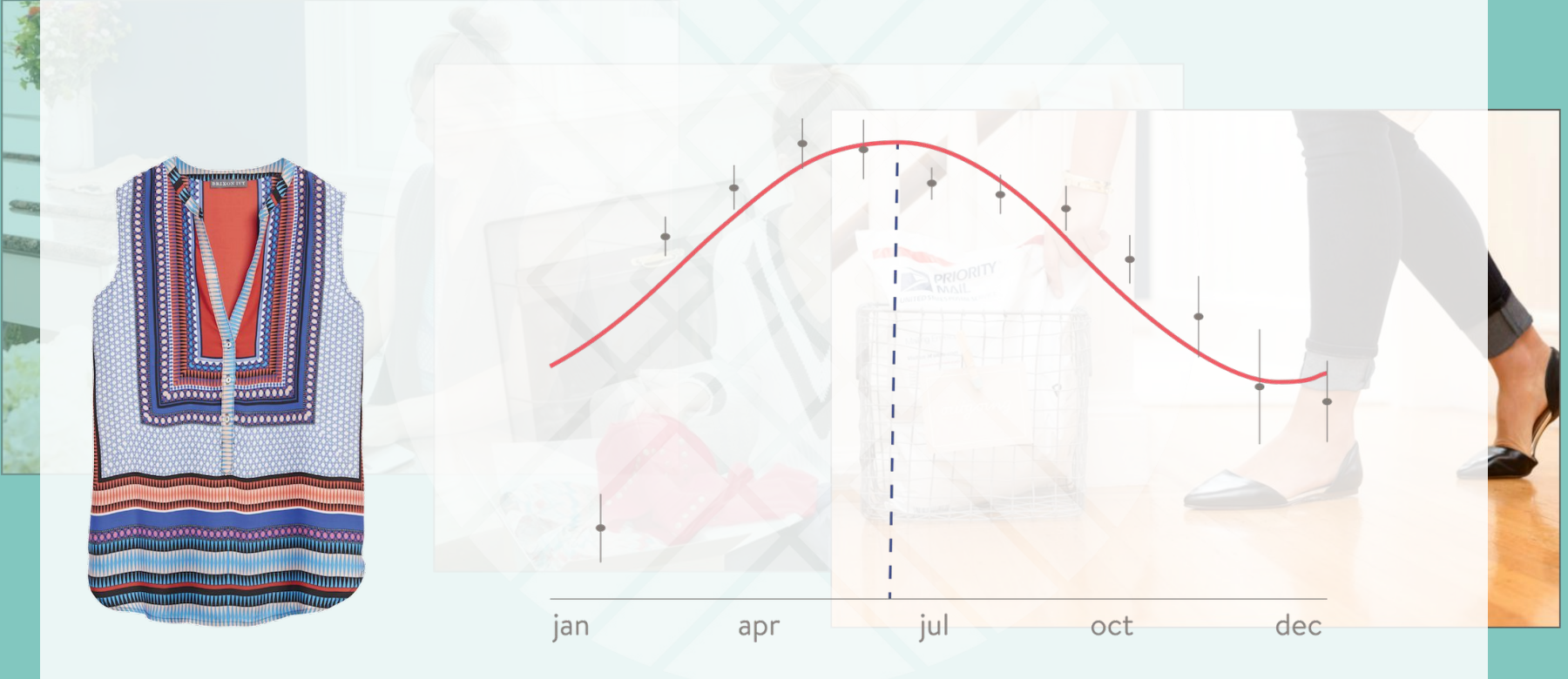
STITCH FIX



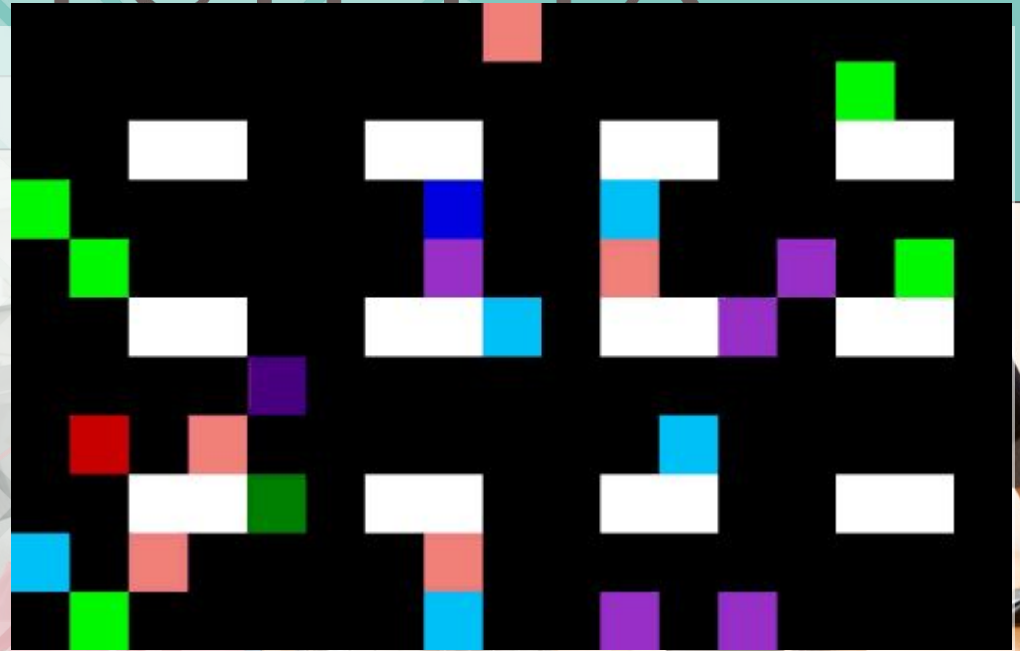
STITCH FIX



STITCH FIX



STITCH FIX



jan

apr

jul

oct

dec



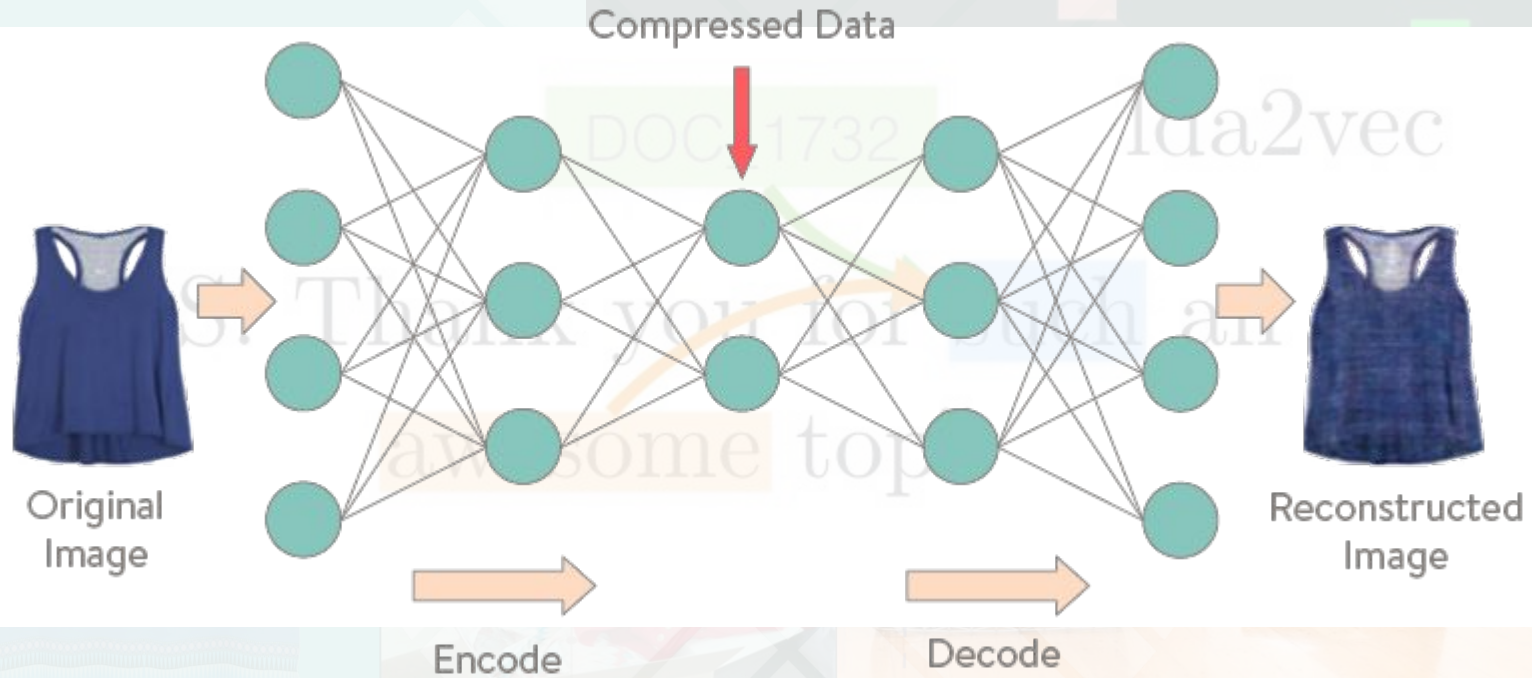
STITCH FIX

DOC_1732

lda2vec

“PS! Thank you for **such** an
awesome top”

STITCH FIX



Two Data Scientist facts:

1. Has AWS console access*.
2. End to end,
they're responsible.

How do we enable this without

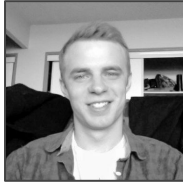


Make doing the *right*
thing the *easy* thing.

Fellow Collaborators



jeff



thomas



akshay



jacob



kurt



derek



patrick



tarek

Horizontal team focused on Data Scientist Enablement

1. Eng. Skills
2. Important
3. What they work on

Let's Start

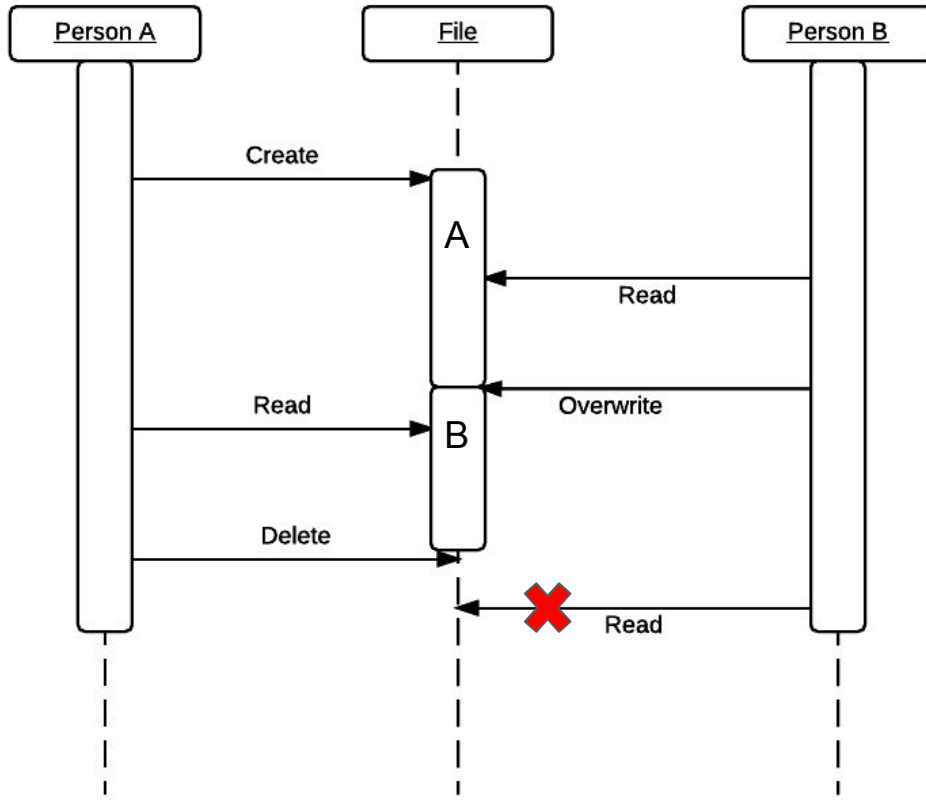
Will Only Cover

1. Source of truth: S3 & Hive Metastore
2. Docker Enabled DS @ Stitch Fix
3. Scaling DS doing ML in the Cloud

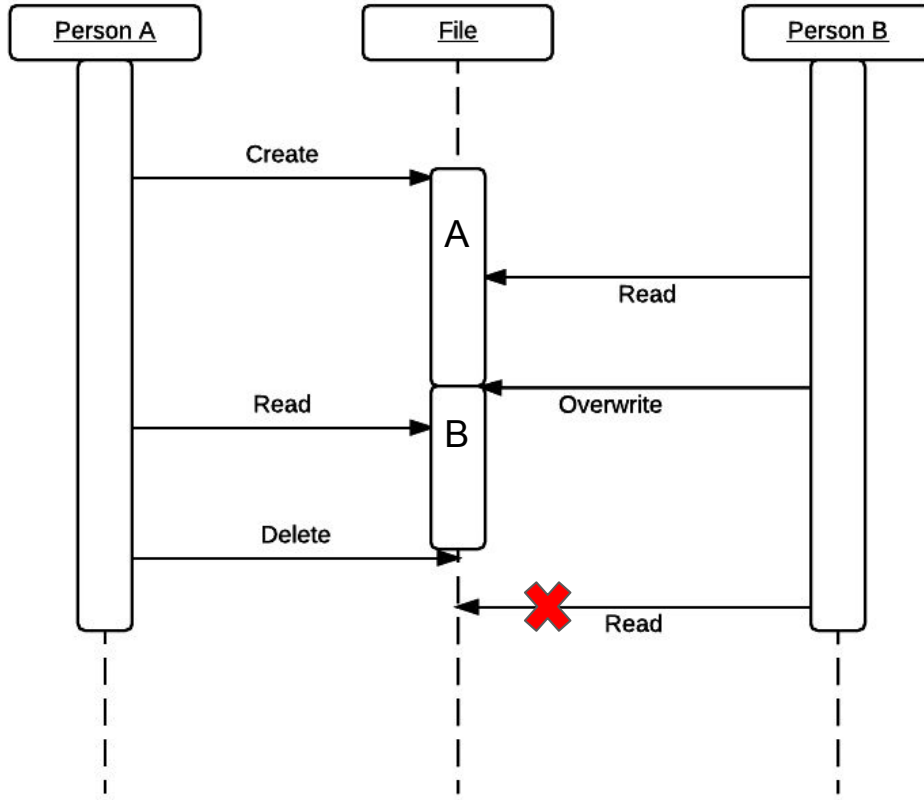
Source of truth:

S3 & Hive Metastore

Want Everyone to Have Same View

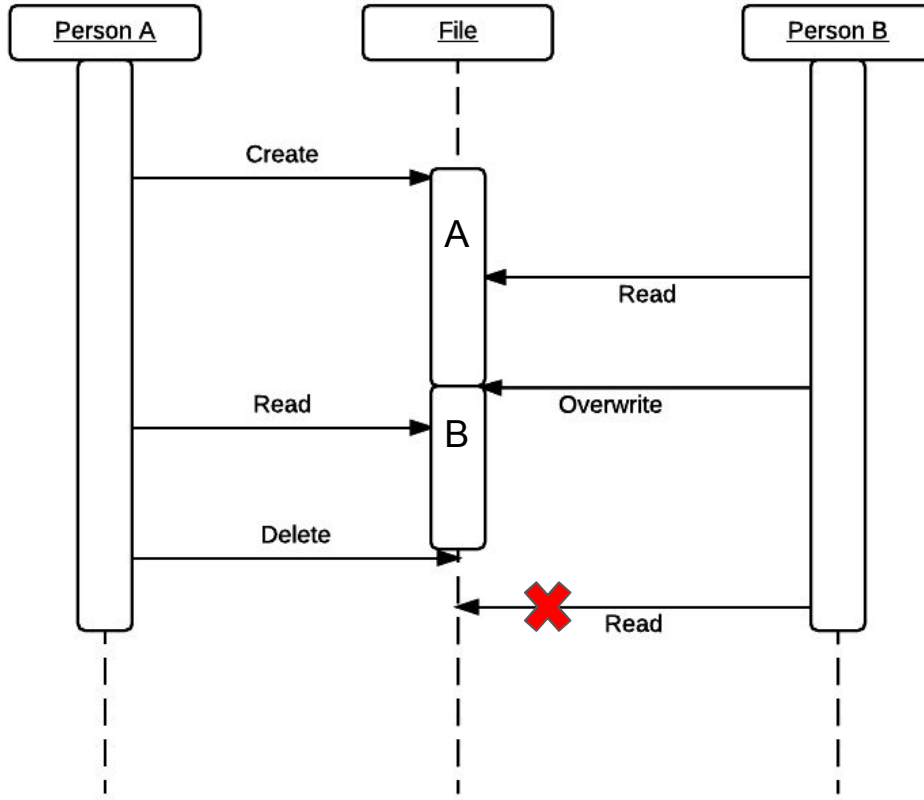


This is Usually Nothing to Worry About



- OS handles correct access
 - DB has ACID properties
-

This is Usually Nothing to Worry About



- OS handles correct access
 - DB has ACID properties
 - But it's easy to outgrow these options with a big data/team.
-

S3

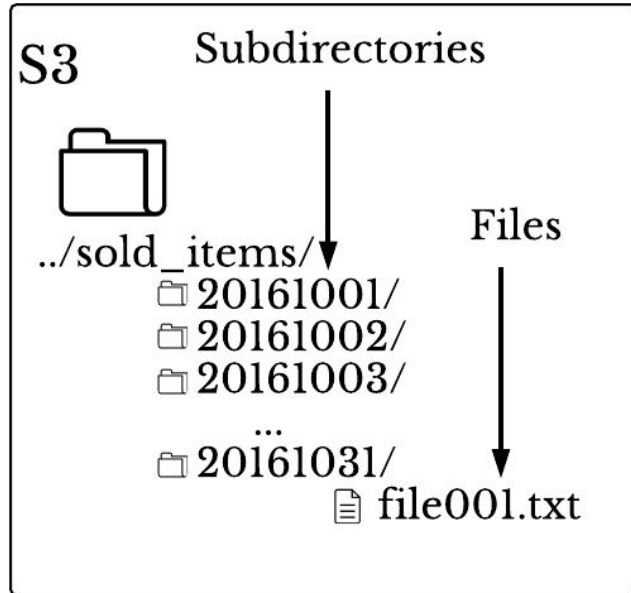
- Amazon's Simple Storage Service
- Infinite* storage
- Can write, read, delete, BUT NOT append.
- Looks like a file system*:
 - URIs: `my.bucket/path/to/files/file.txt`
- Scales well

Hive Metastore

- Hadoop service, that stores:
 - Schema
 - Partition information, e.g. date
 - Data location for a partition

Hive Metastore

- Hadoop service, that stores:
 - Schema
 - Partition information, e.g. date
 - Data location for a partition

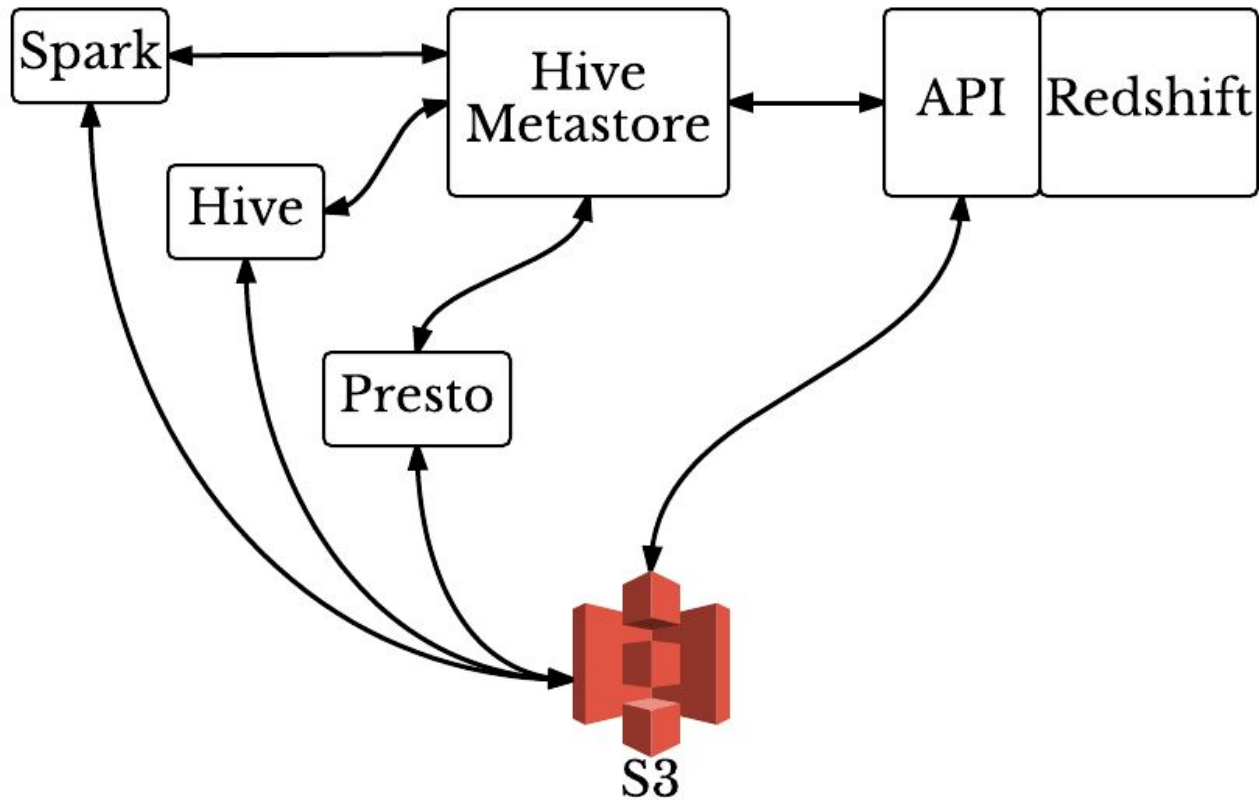


Hive Metastore:

sold_items

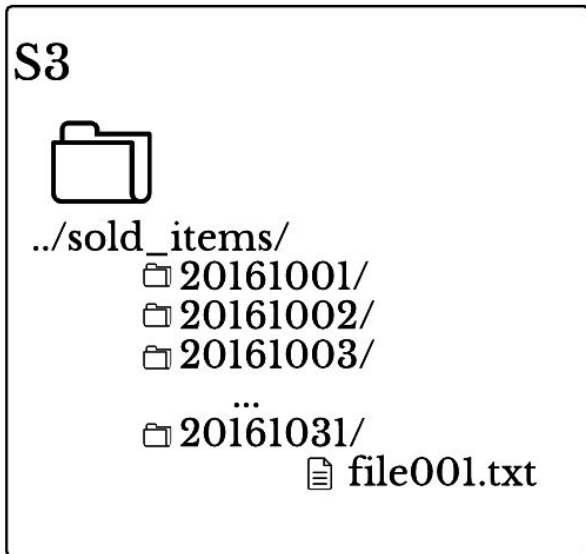
| Partition | Location |
|-----------|---------------------------------|
| 20161001 | s3://bucket/sold_items/20161001 |
| ... | |
| 20161031 | s3://bucket/sold_items/20161031 |

Hive Metastore



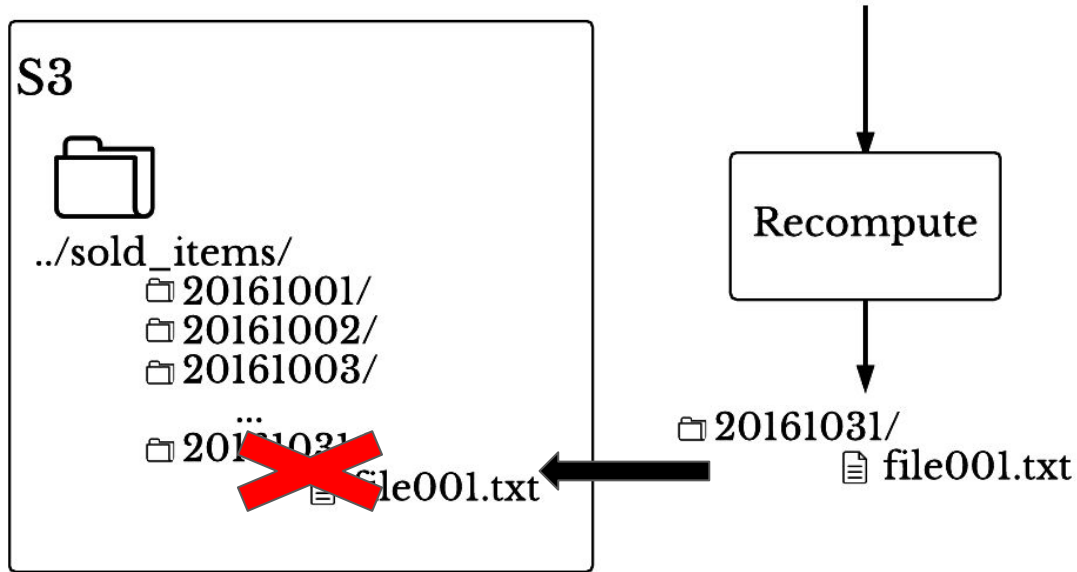
But if we're not careful

- Replacing data in a partition

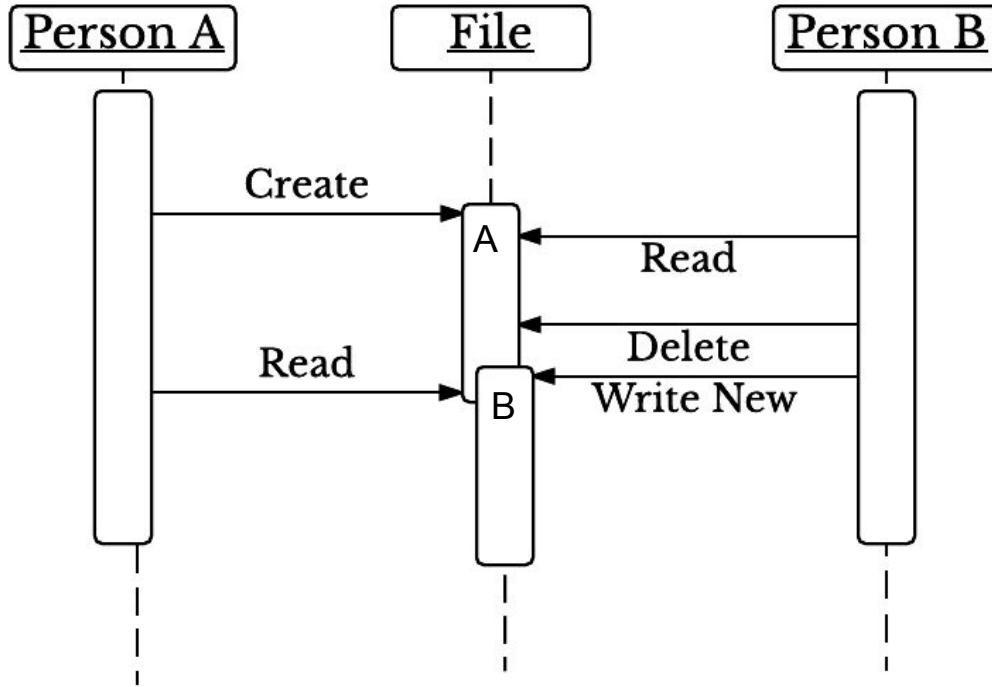


But if we're not careful

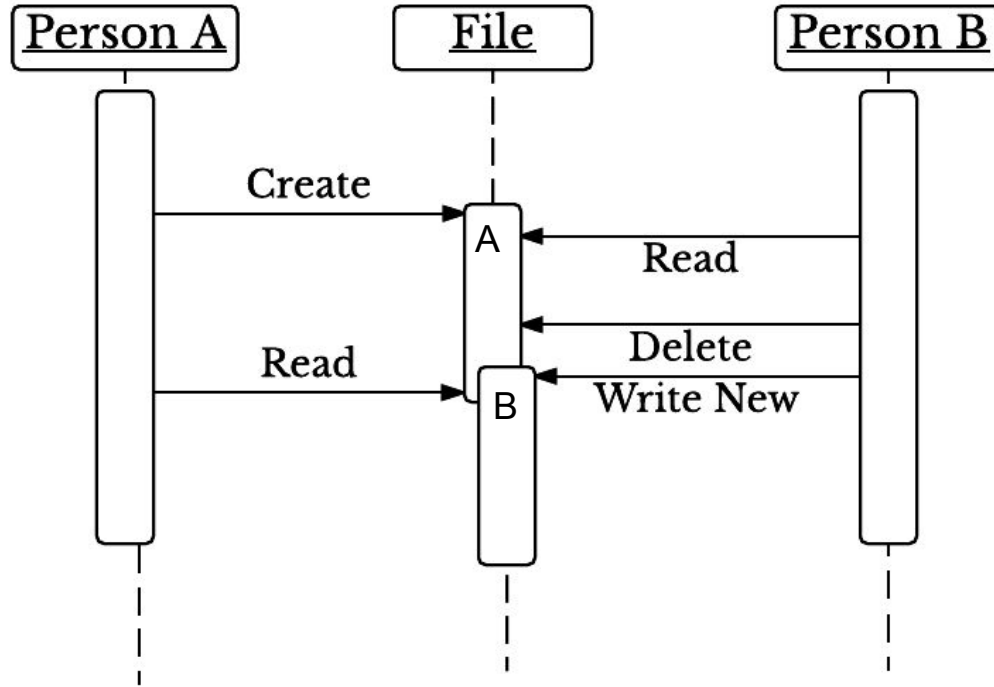
- Replacing data in a partition



But if we're not careful



But if we're not careful

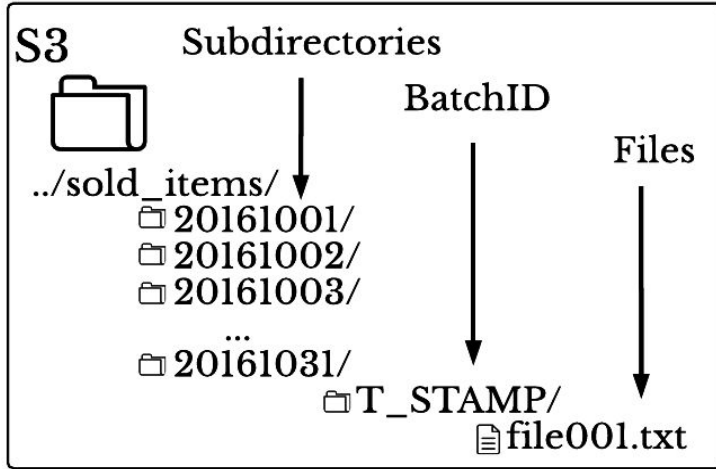


- S3 is eventually consistent
 - These bugs are hard to track down
-

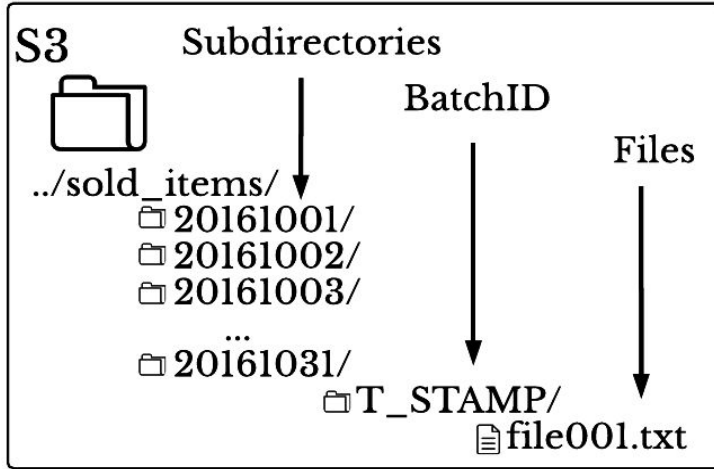
Hive Metastore to the Rescue

- Use Hive Metastore to control partition source of truth
- Principles:
 - Never delete
 - Always write to a *new* place each time a partition changes
- Stitch Fix solution:
 - Use an inner directory → called Batch ID

Batch ID Pattern



Batch ID Pattern

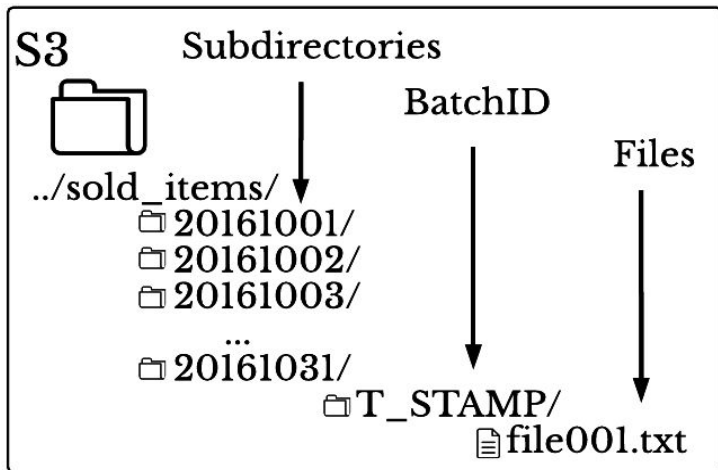


sold_items

| Date | Location |
|----------|---|
| 20161001 | s3://bucket/sold_items/ 20161001/20161002002334/ |
| ... | ... |
| 20161031 | s3://bucket/sold_items/ 20161031/20161101002256/ |

Batch ID Pattern

- Overwriting a partition is just a matter of updating the location

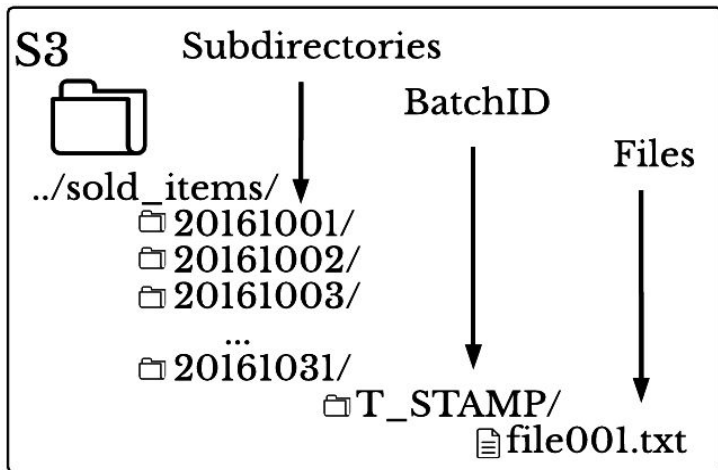


`sold_items`

| Date | Location |
|----------|---|
| 20161001 | <code>s3://bucket/sold_items/ 20161001/20161002002334/</code> |
| ... | ... |
| 20161031 | <code>s3://bucket/sold_items/ 20161031/20161101002256/ s3://bucket/sold_items/ 20161031/20161102234252</code> |

Batch ID Pattern

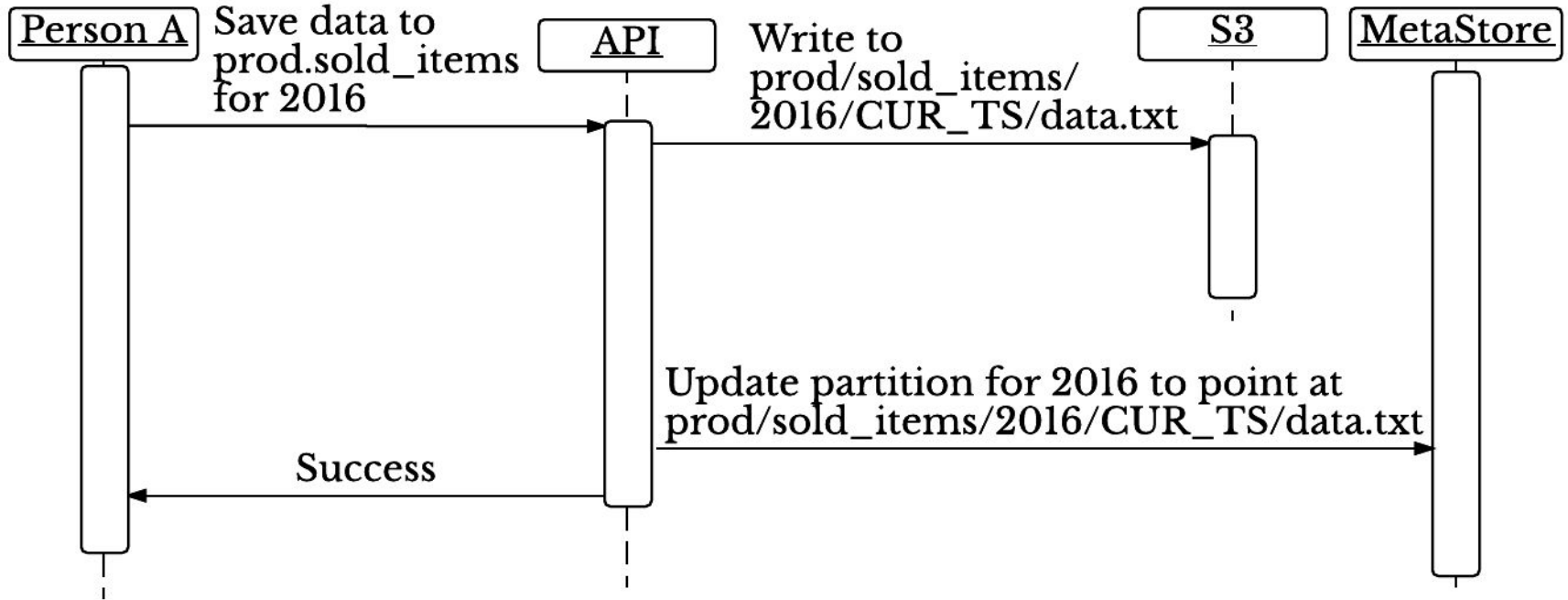
- Overwriting a partition is just a matter of updating the location
- To the user this is a *hidden inner* directory



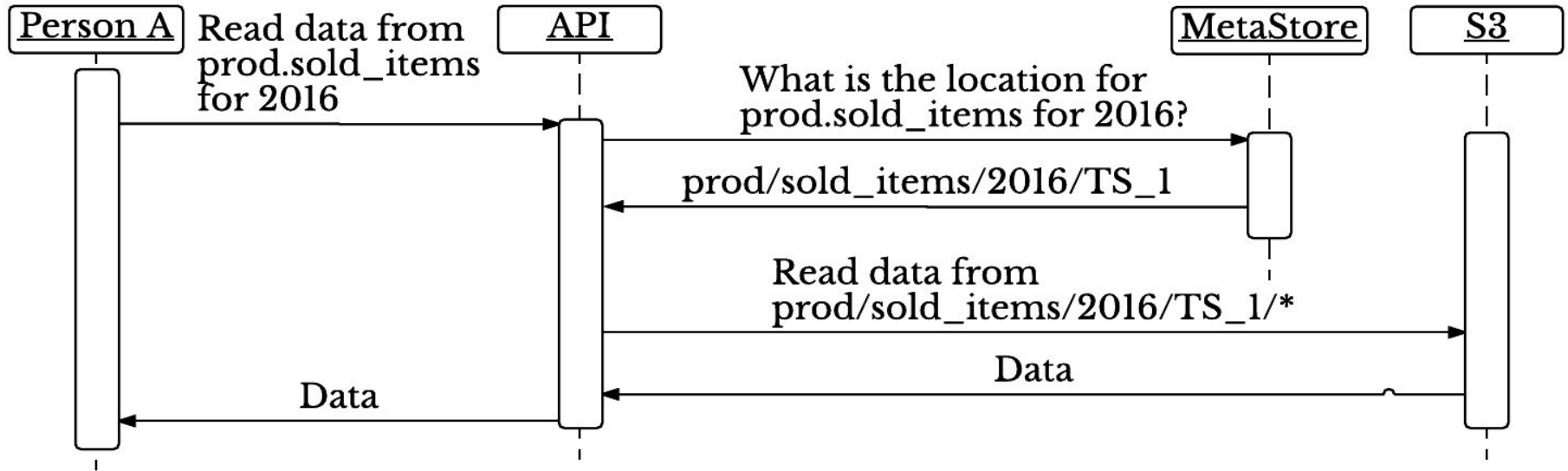
`sold_items`

| Date | Location |
|----------|---|
| 20161001 | <code>s3://bucket/sold_items/ 20161001/20161002002334/</code> |
| ... | ... |
| 20161031 | <code>s3://bucket/sold_items/ 20161031/20161101002256/ s3://bucket/sold_items/ 20161031/20161102234252</code> |

Enforce via API



Enforce via API



API for Data Scientists

Python:

```
store_dataframe(df, dest_db, dest_table, partitions=['2016'])  
df = load_dataframe(src_db, src_table, partitions=['2016'])
```

R:


```
sf_writer(data      = result,  
          namespace = dest_db,  
          resource  = dest_table,  
          partitions = c(as.integer(opt$ETL_DATE)))  
  
sf_reader(namespace = src_db,  
          resource  = src_table,  
          partitions = c(as.integer(opt$ETL_DATE)))
```

Batch ID Pattern Benefits



- Full partition history
 - Can rollback
 - Data Scientists are less afraid of mistakes
 - Can create audit trails more easily
 - What data changed and when
 - Can anchor downstream consumers to a particular batch ID

Docker Enabled DS @ Stitch Fix




Ad hoc Infra: In the Beginning...

| Workstation | Env. Mgmt. | Scalability |
|---|------------|-------------|
|  | Low | Low |
| | | |
| | | |



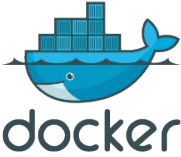
Ad hoc Infra: Evolution I

| Workstation | Env. Mgmt. | Scalability |
|---|------------|-------------|
|  | Low | Low |
|  | Medium | Medium |
| | | |

Ad hoc Infra: Evolution II

| Workstation | Env. Mgmt. | Scalability |
|---|------------|-------------|
|  | Low | Low |
|  | Medium | Medium |
|  | High | High |

Ad hoc Infra: Evolution III

| Workstation | Env. Mgmt. | Scalability |
|---|------------|-------------|
|  | Low | Low |
|  | Medium | Medium |
|  | Low | High |

Why Does Docker Lower Overhead?

- Control of environment
 - Data Scientists don't need to worry about env.
- Isolation
 - can host many docker containers on a single machine.
- Better host management
 - allowing central control of machine types.

Flotilla UI

Add a compute container

Alias

Memory

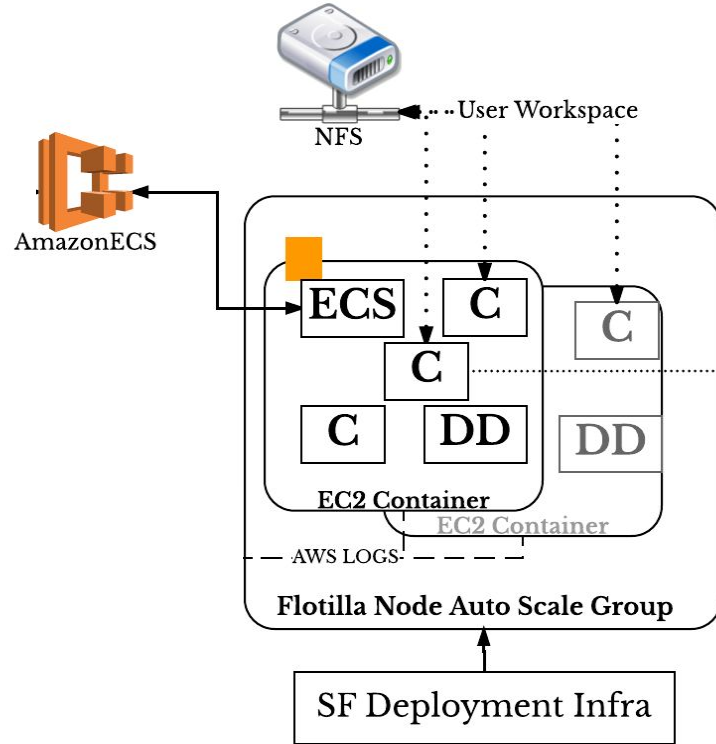
1 Active Containers

| name / alias | version | jupyter | rstudio | status | memory | uptime | \$ so far | actions |
|---|---------------|--|--|----------------|--------|---------------|-----------|----------------------------------|
| stefan_qcon_sf <input type="text"/> | /flotilla:1.2 | <input type="button" value="jupyter"/> | <input type="button" value="rstudio"/> | RUNNING | 4 GB | a few seconds | \$0 | <input type="button" value="✕"/> |

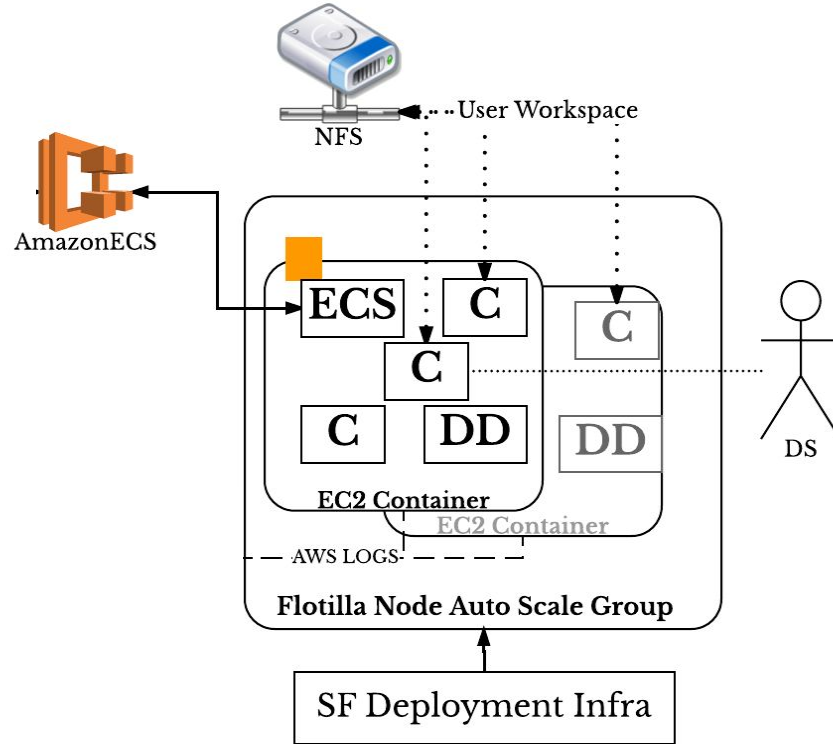
Our Docker Image

- Has:
 - Our internal API libraries
 - Jupyter Notebook:
 - Pyspark
 - IPython
 - Python libs:
 - scikit, numpy, scipy, pandas, etc.
 - RStudio
 - R libs:
 - Dplyr, magrittr, ggplot2, lme4, BOOT, etc.
 - Mounts User NFS
 - User has terminal access to file system via Jupyter for git, pip, etc.
-

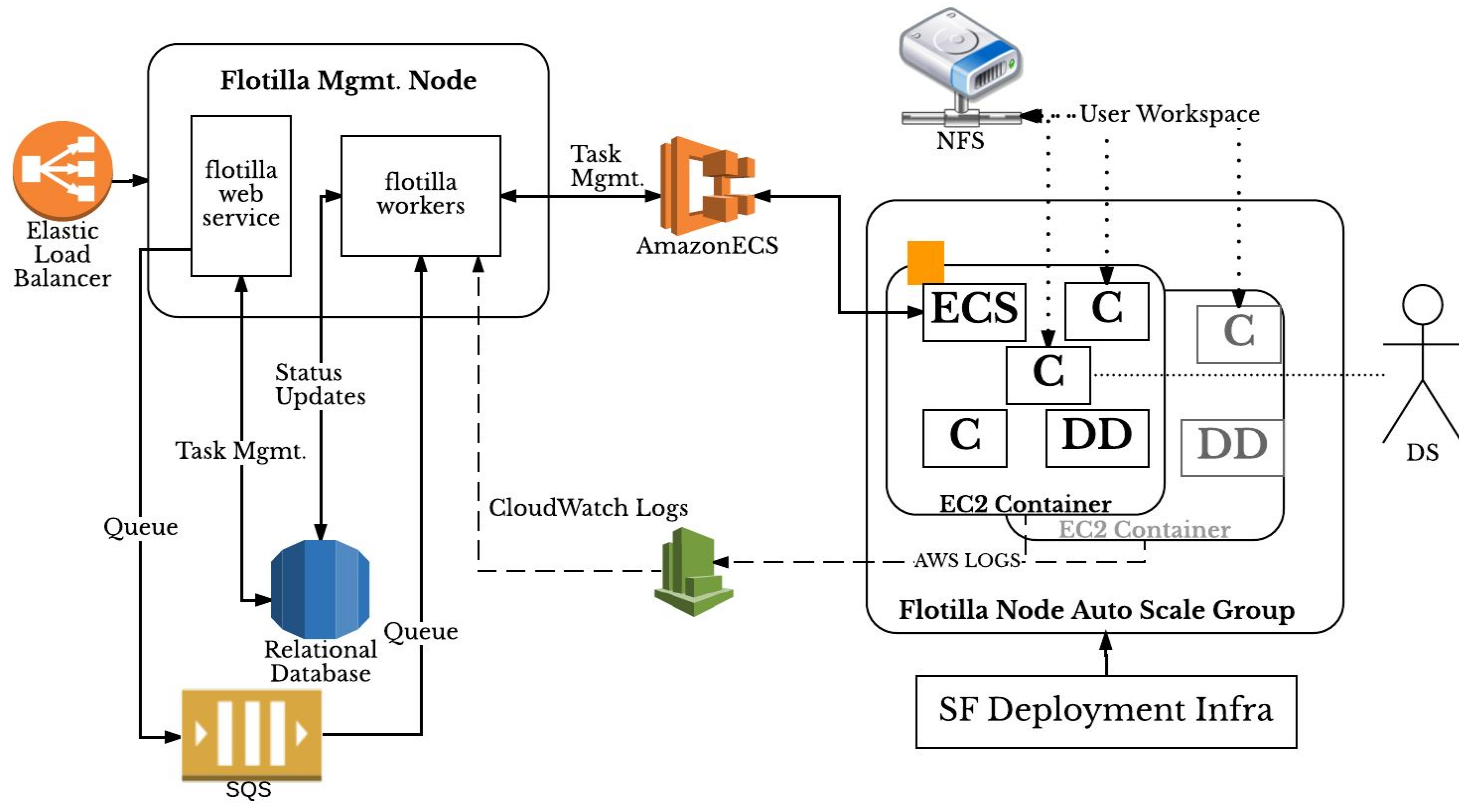
Docker Deployment



Docker Deployment



Docker Deployment



Our Docker Problems So Far

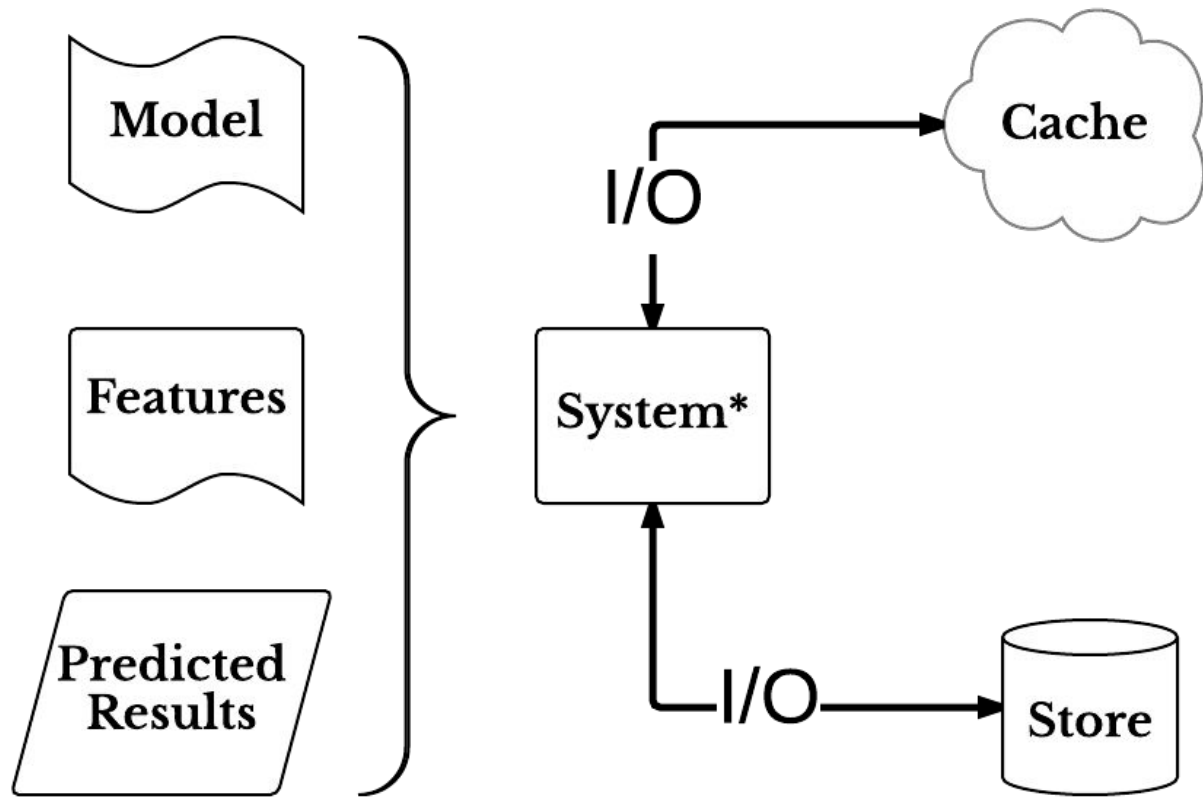
- Docker tightly integrates with the Linux Kernel.
 - Hypothesis:
 - Anything that makes uninterruptable calls to the kernel can:
 - Break the ECS agent because the container doesn't respond.
 - Break isolation between containers.
 - E.g. Mounting NFS
- Docker Hub:
 - Switched to artifactory

Scaling DS doing ML in the Cloud

1. Data Latency
2. To Batch
or Not To Batch
3. What's in a Model?

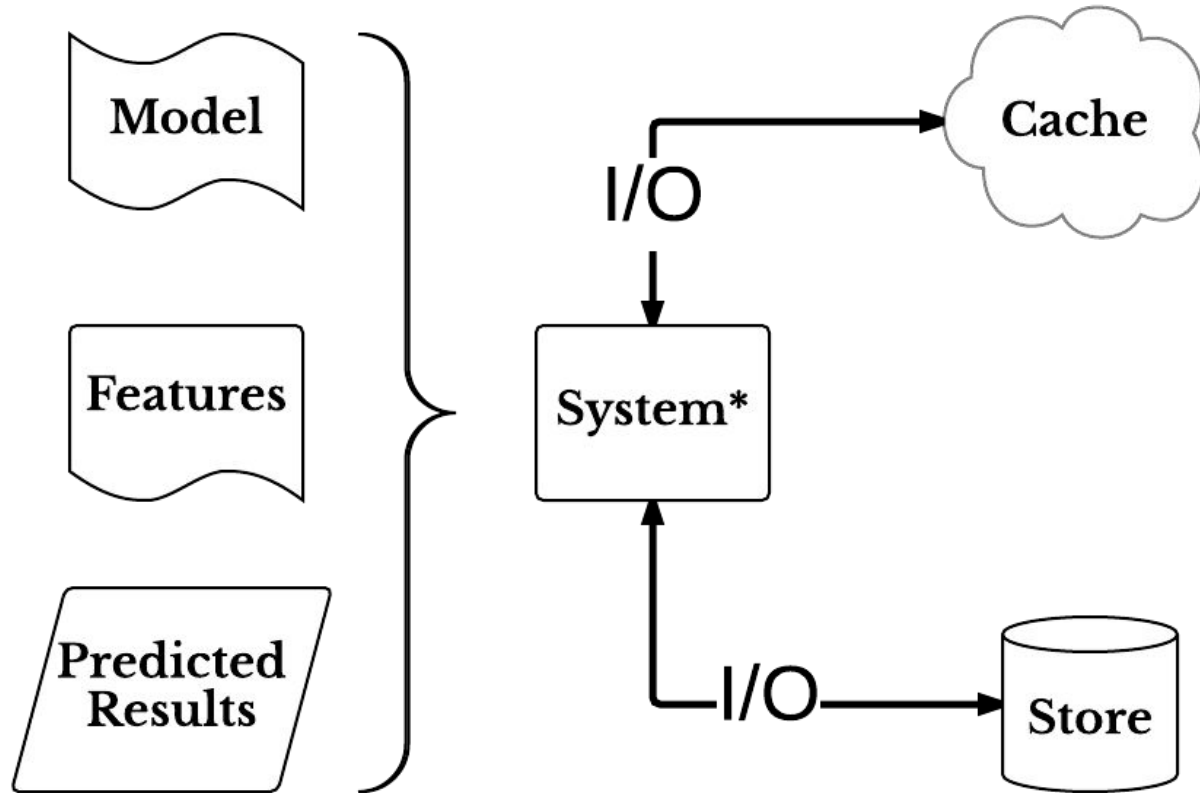
Data Latency

How much time do you spend waiting for data?



*This could be a laptop, a shared system, a batch process, etc.

Use Compression



*This could be a laptop, a shared system, a batch process, etc.

Use Compression - The Components

Model

[1.3234543 0.23443434 ...]
[1 0 0 1 0 0 ... 0 1 0 0
 0 1 0 1 ...
 ... 1 0 1 1]

Features

[1 0 0 1 0 0 ... 0 1 0 0]
[1.3234543 0.23443434 ...]

**Predicted
Results**

{ 100: 0.56, ... , 110: 0.65,
 ... , ... , 999: 0.43 }

Use Compression - Python Comparison

Model

```
[1.3234543 0.23443434 ... ]  
[1 0 0 1 0 0 ...      0 1 0 0  
 0 1 0 1 ...  
                                ...      1 0 1 1]
```

Pickle: 60MB
Zlib+Pickle: 129KB
JSON: 15MB
Zlib+JSON: 55KB

Features

```
[1 0 0 1 0 0 ... 0 1 0 0 ]  
[1.3234543 0.23443434 ... ]
```

Pickle: 3.1KB
Zlib+Pickle: 921B
JSON: 2.8KB
Zlib+JSON: 681B

**Predicted
Results**

```
{ 100: 0.56, ... ,110: 0.65,  
  ... , ... , 999: 0.43 }
```

Pickle: 2.6MB
Zlib+Pickle: 600KB
JSON: 769KB
Zlib+JSON: 139KB

Observations

- Naïve scheme of JSON + Zlib works well:

```
import json
import zlib
...
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```

Observations

- Naïve scheme of JSON + Zlib works well:

```
import json
import zlib
...
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```

- Double vs Float: do you really need to store that much precision?
-

Observations

- Naïve scheme of JSON + Zlib works well:

```
import json
import zlib
...
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```

- Double vs Float: do you really need to store that much precision?
 - For more inspiration look to columnar DBs and how they compress columns
-

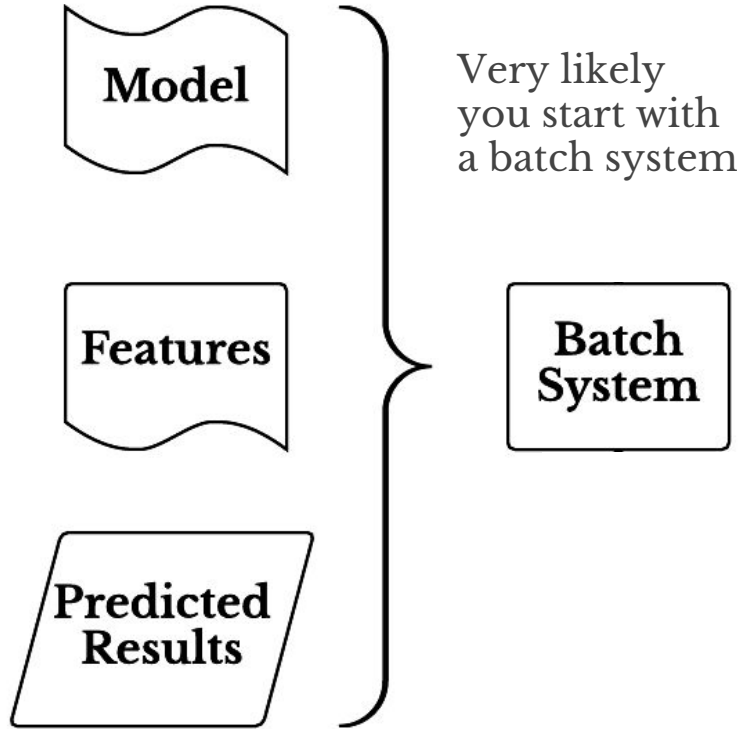
To Batch or Not To Batch:

When is batch inefficient?

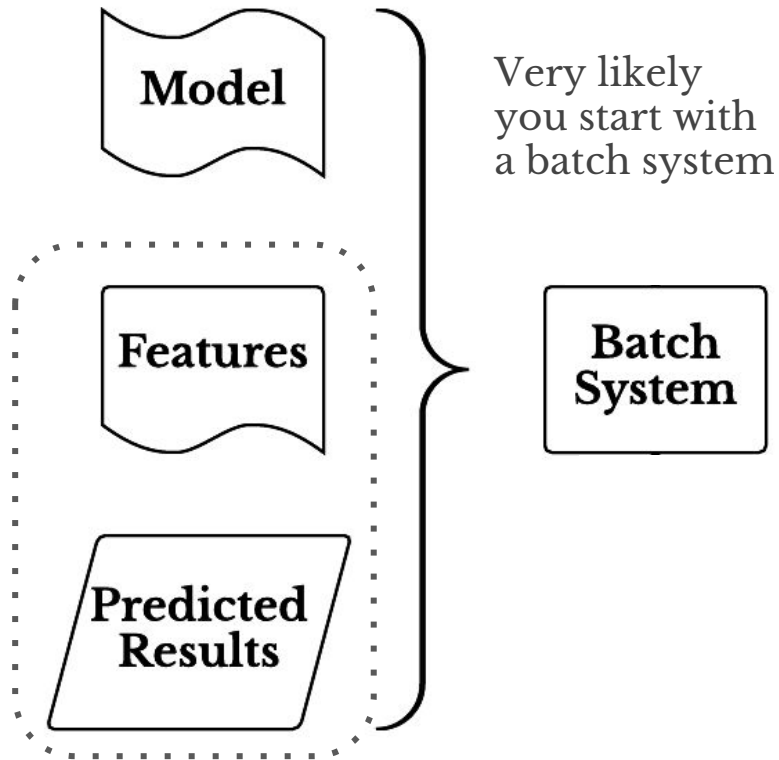
Online & Streamed Computation

- Online:
 - Computation occurs synchronously when needed.
- Streamed:
 - Computation is triggered by an event(s).

Online & Streamed Computation

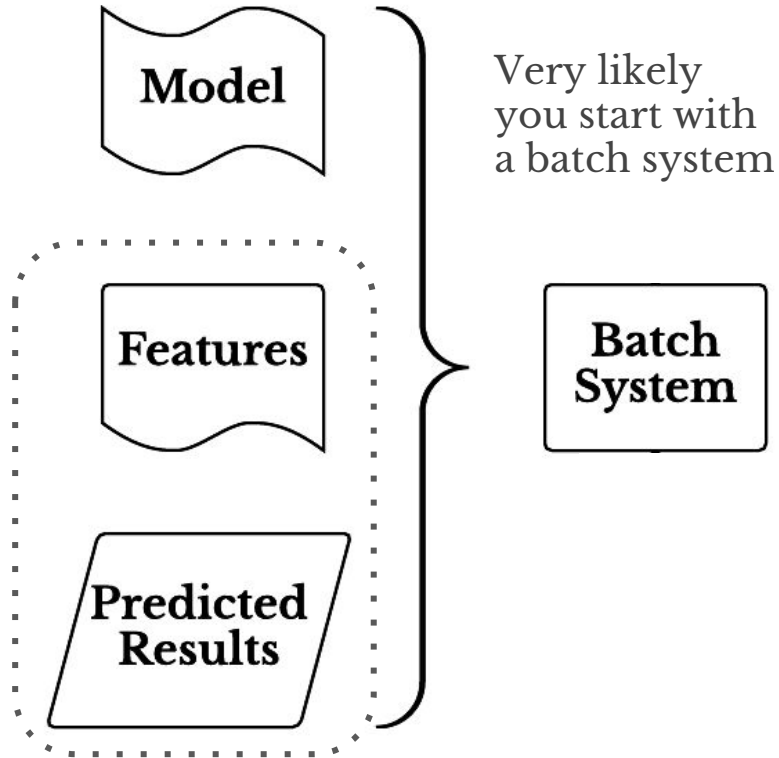


Online & Streamed Computation



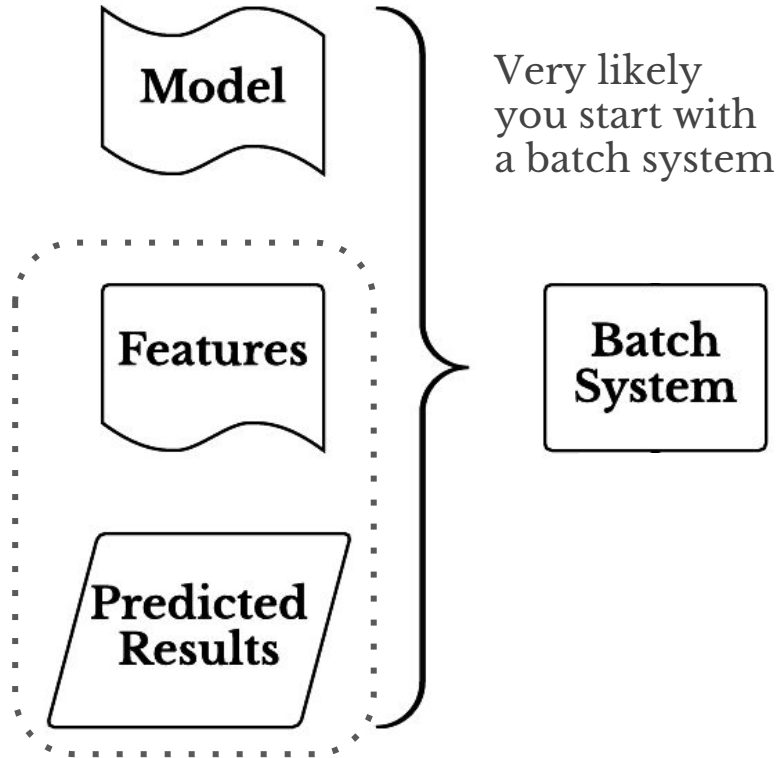
- Do you need to recompute:
 - features for all users?
 - predicted results for all users?

Online & Streamed Computation



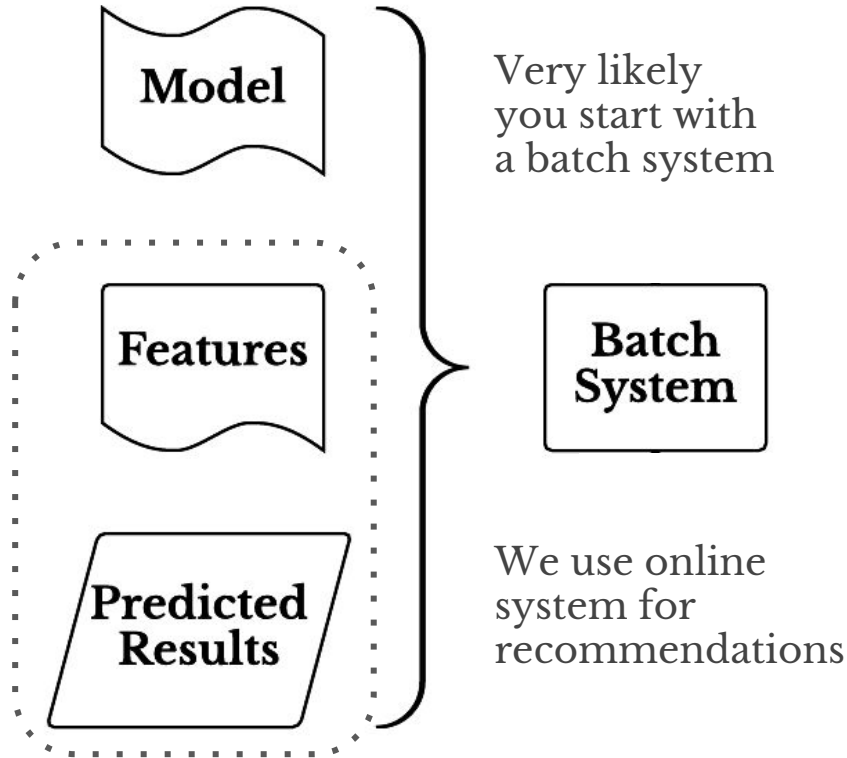
- Do you need to recompute:
 - features for all users?
 - predicted results for all users?
- Are you heavily dependent on your ETL running every night?

Online & Streamed Computation



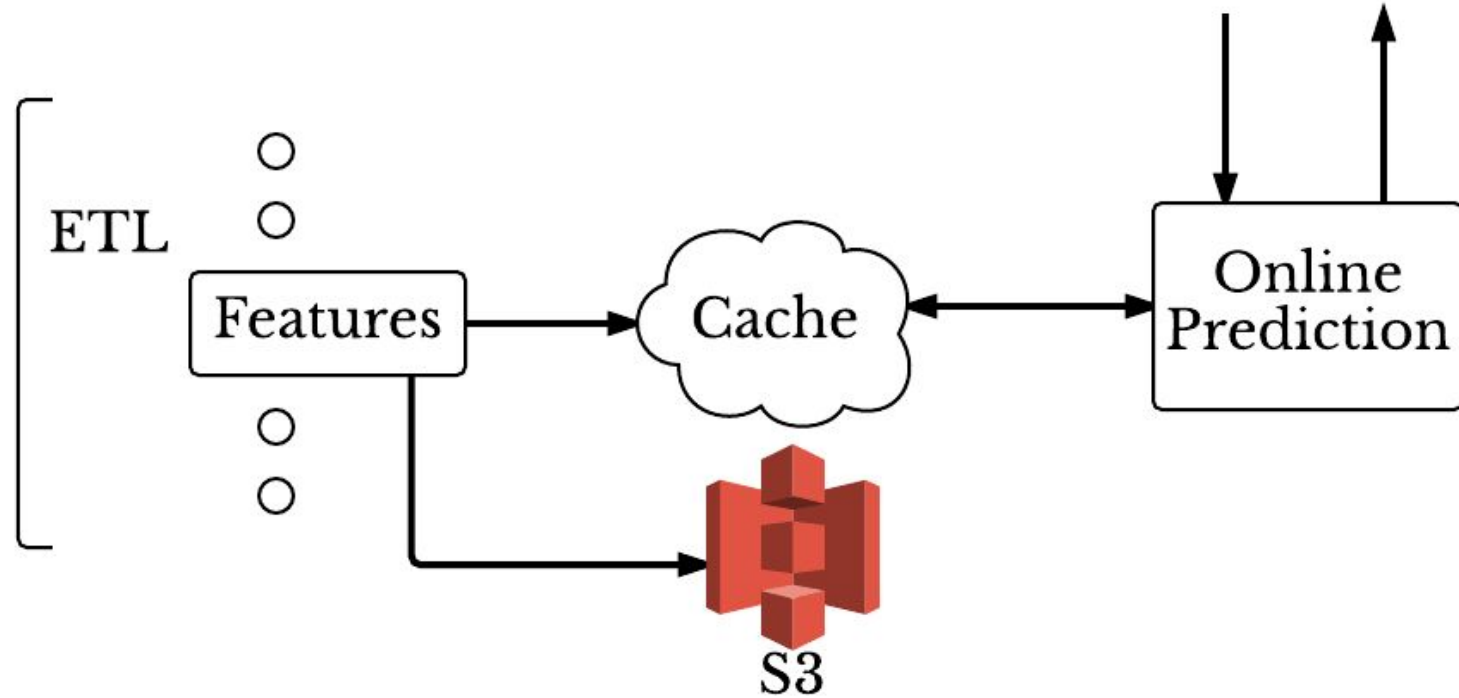
- Do you need to recompute:
 - features for all users?
 - predicted results for all users?
- Are you heavily dependent on your ETL running every night?
- Online vs Streamed depends on in house factors:
 - Number of models
 - How often they change
 - Cadence of output required
 - In house eng. expertise
 - etc.

Online & Streamed Computation

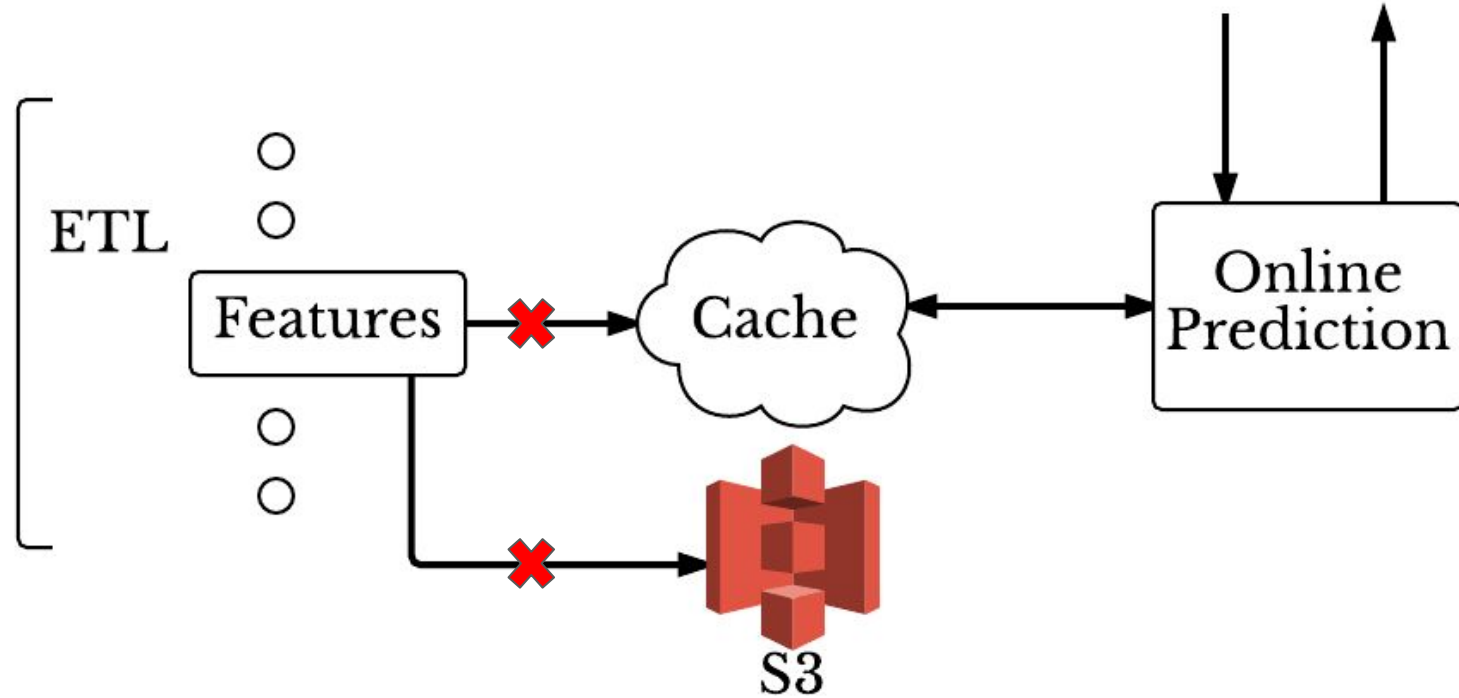


- Do you need to recompute:
 - features for all users?
 - predicted results for all users?
- Are you heavily dependent on your ETL running every night?
- Online vs Streamed depends on in house factors:
 - Number of models
 - How often they change
 - Cadence of output required
 - In house eng. expertise
 - etc.

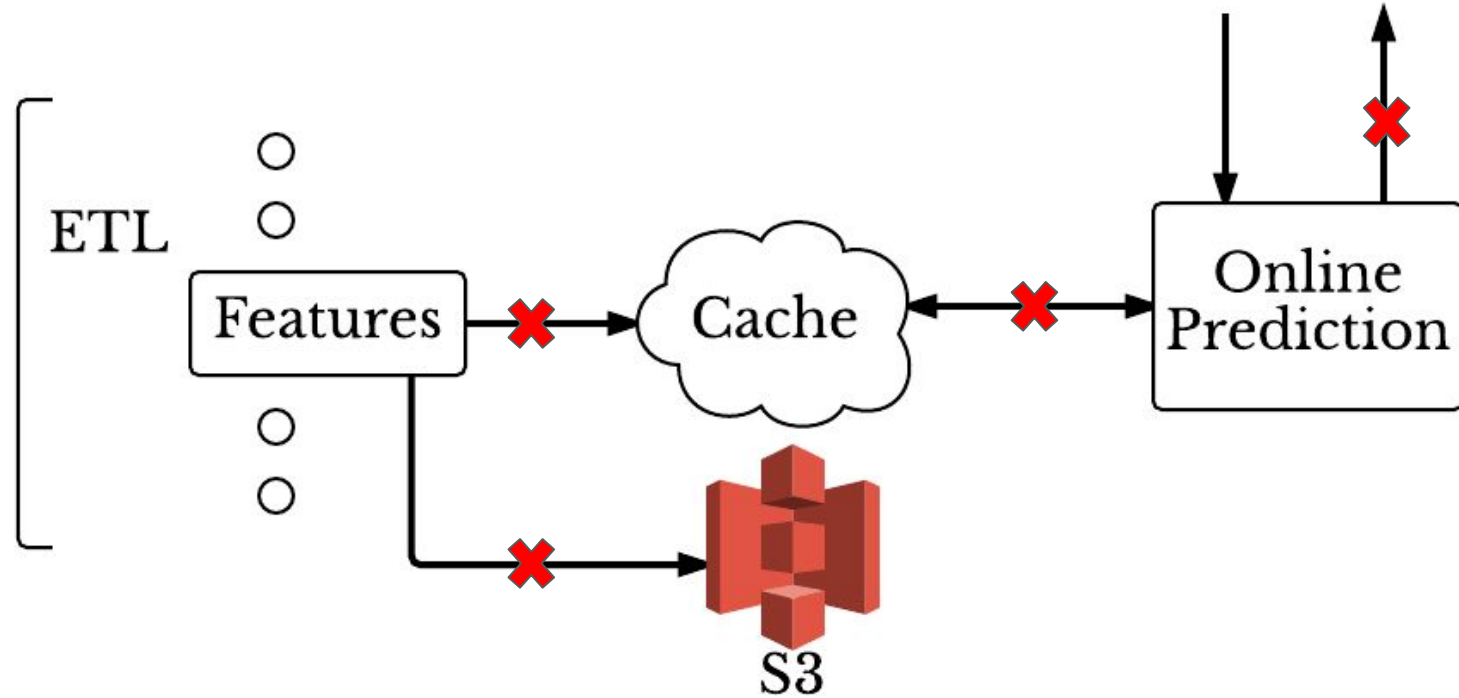
Streamed Example



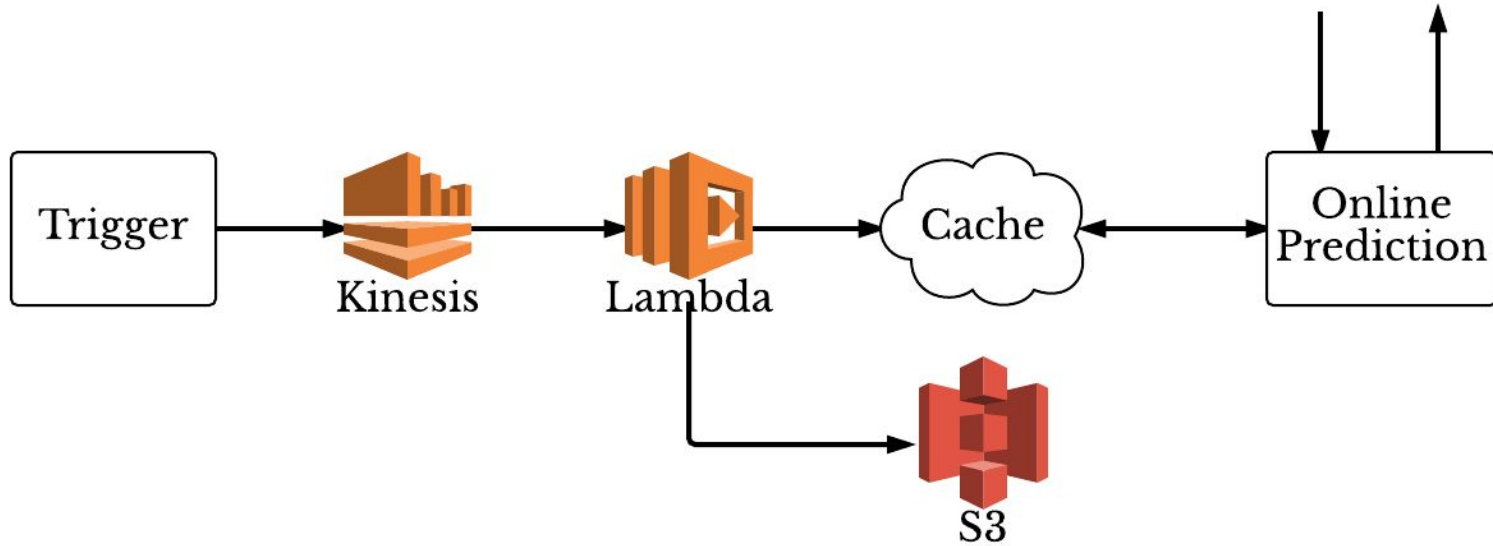
Streamed Example



Streamed Example



Streamed Example



Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
 - Hopefully \$\$\$ savings
 - Hopefully less stressed Data Scientists

Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
 - Hopefully \$\$\$ savings
 - Hopefully less stressed Data Scientists
- Requires better software engineering practices
 - Code portability/reuse
 - Designing APIs/Tools Data Scientists will use

Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
 - Hopefully \$\$\$ savings
 - Hopefully less stressed Data Scientists
- Requires better software engineering practices
 - Code portability/reuse
 - Designing APIs/Tools Data Scientists will use
- Prototyping on AWS Lambda & Kinesis was surprisingly quick
 - Need to compile C libs on an amazon linux instance

What's in a Model?

Scaling model knowledge

Ever:

- Had someone leave and then nobody understands how they trained their models?

Ever:

- Had someone leave and then nobody understands how they trained their models?
 - Or you didn't remember yourself?

Ever:

- Had someone leave and then nobody understands how they trained their models?
 - Or you didn't remember yourself?
- Had performance dip in models and you have trouble figuring out why?

Ever:

- Had someone leave and then nobody understands how they trained their models?
 - Or you didn't remember yourself?
- Had performance dip in models and you have trouble figuring out why?
 - Or not known what's changed between model deployments?

Ever:

- Had someone leave and then nobody understands how they trained their models?
 - Or you didn't remember yourself?
- Had performance dip in models and you have trouble figuring out why?
 - Or not known what's changed between model deployments?
- Wanted to compare model performance over time?

Ever:

- Had someone leave and then nobody understands how they trained their models?
 - Or you didn't remember yourself?
 - Had performance dip in models and you have trouble figuring out why?
 - Or not known what's changed between model deployments?
 - Wanted to compare model performance over time?
 - Wanted to train a model in R/Python/Spark and then deploy it a webserver?
-

Produce Model Artifacts

Produce Model Artifacts

- Isn't that just saving the coefficients/model values?

Produce Model Artifacts

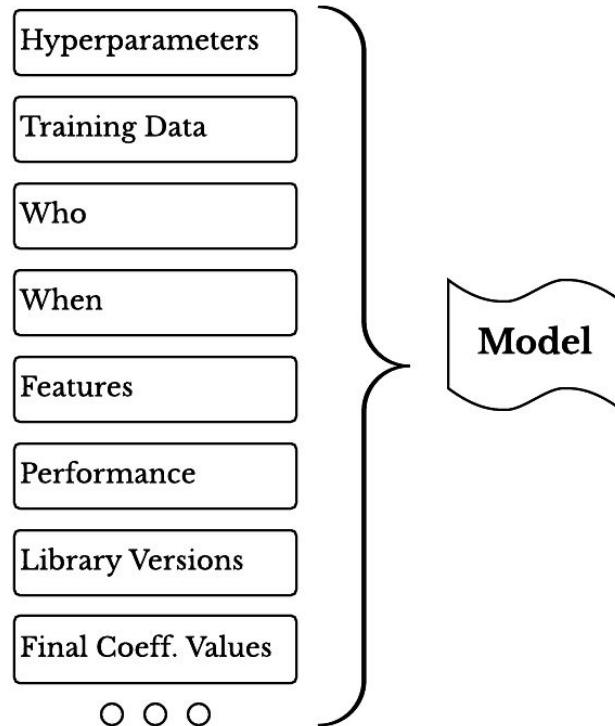
- Isn't that just saving the coefficients/model values?
 - NO!

Produce Model Artifacts

- Isn't that just saving the coefficients/model values?
 - NO!
- Why?

Produce Model Artifacts

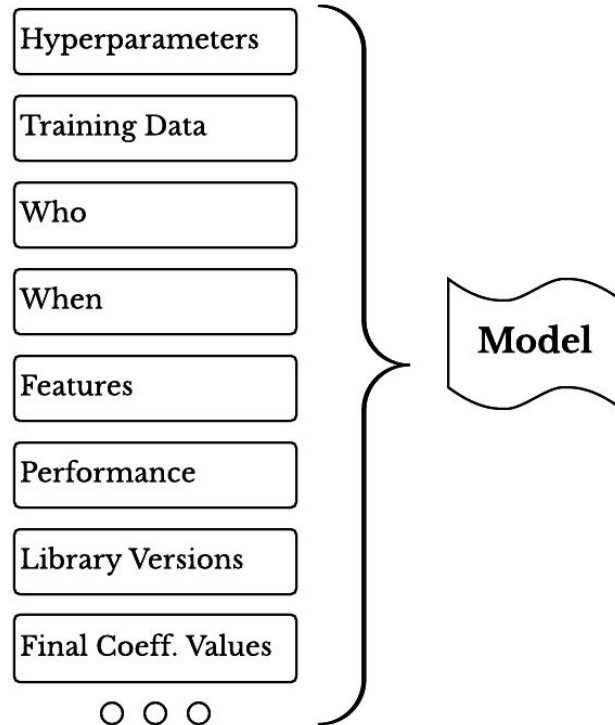
- Isn't that just saving the coefficients/model values?
 - NO!
- Why?



Produce Model Artifacts

- Isn't that just saving the coefficients/model values?
 - NO!
- Why?

How do you deal with organizational drift?

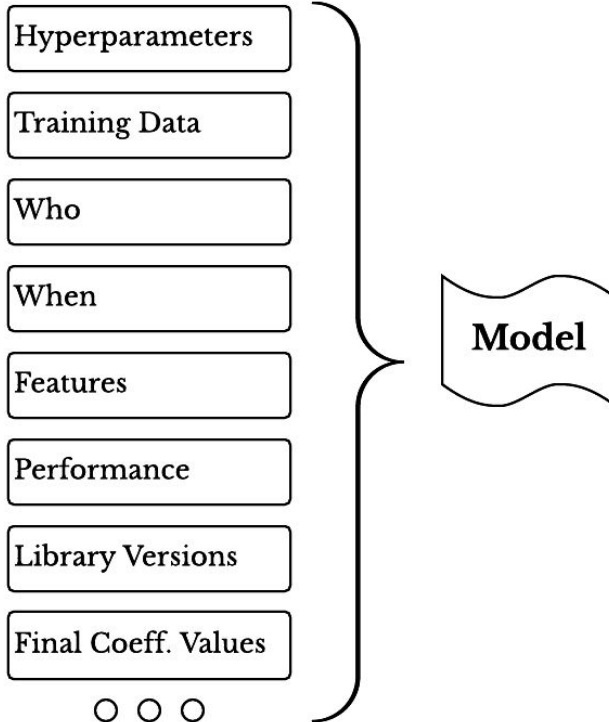


Produce Model Artifacts

- Isn't that just saving the coefficients/model values?
 - NO!
- Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time



Produce Model Artifacts

- Isn't that just saving the coefficients/model values?
 - NO!
- Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time

Hyperparameters

Training Data

Who

When

Features

Performance

Library Versions

Final Coeff. Values

○ ○ ○



Model

Helps a lot with model debugging & diagnosis!

Produce Model Artifacts

- Isn't that just saving the coefficients/model values?
 - NO!
- Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time

Hyperparameters

Training Data

Who

When

Features

Performance

Library Versions

Final Coeff. Values



Model

Helps a lot with model debugging & diagnosis!

Can more easily use in downstream processes

Produce Model Artifacts

- Analogous to software libraries
- Packaging:
 - Zip/Jar file

But all the above
seems complex?

We're building APIs.

Fin; Questions?

@stefkrawczyk