Varant Zanoyan / Strata

Zipline – Airbnb's ML Data Management Framework

Varant Zanoyan

Team: Machine Learning Infrastructure
Role: Software Engineer

Agenda

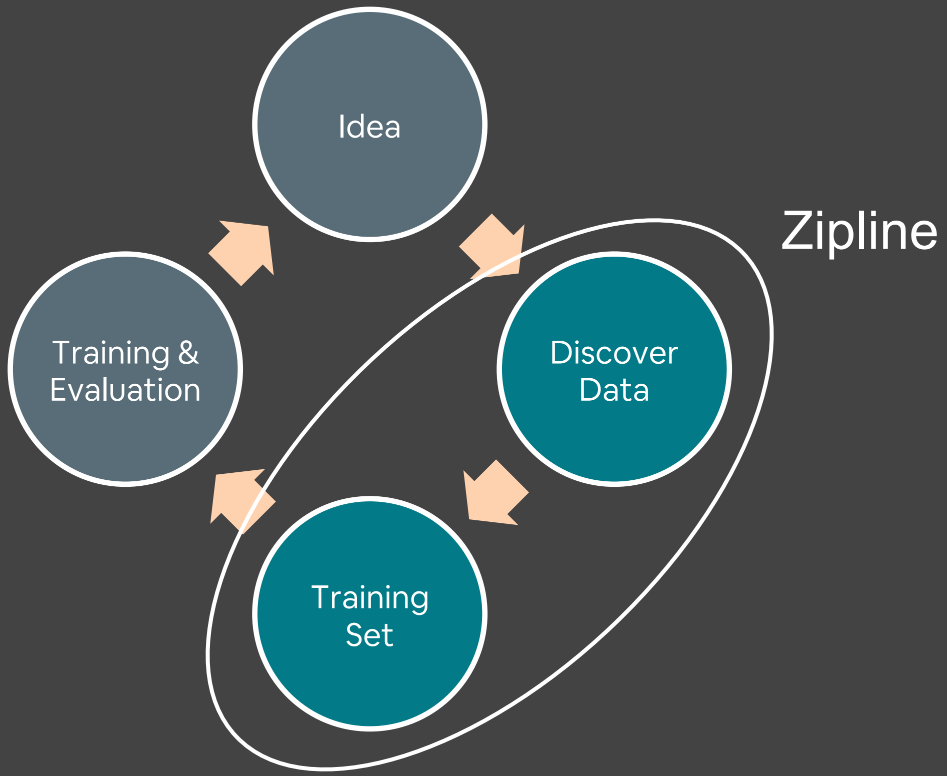
1) The Machine Learning Workflow
2) Motivation for Zipline (the problem)
3) Zipline implementation (the solution)
4) Deep dive (technical)

Team Mission

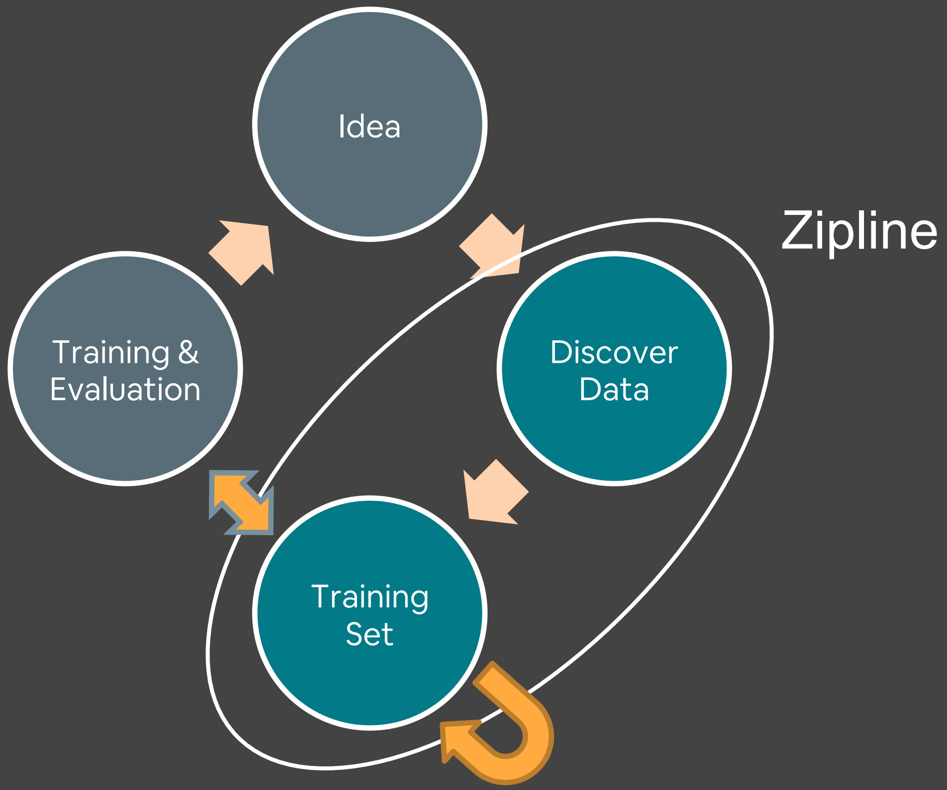
Equip Airbnb with shared technology to build *production-ready* ML applications with no *incidental complexity*.

The Machine Learning Workflow

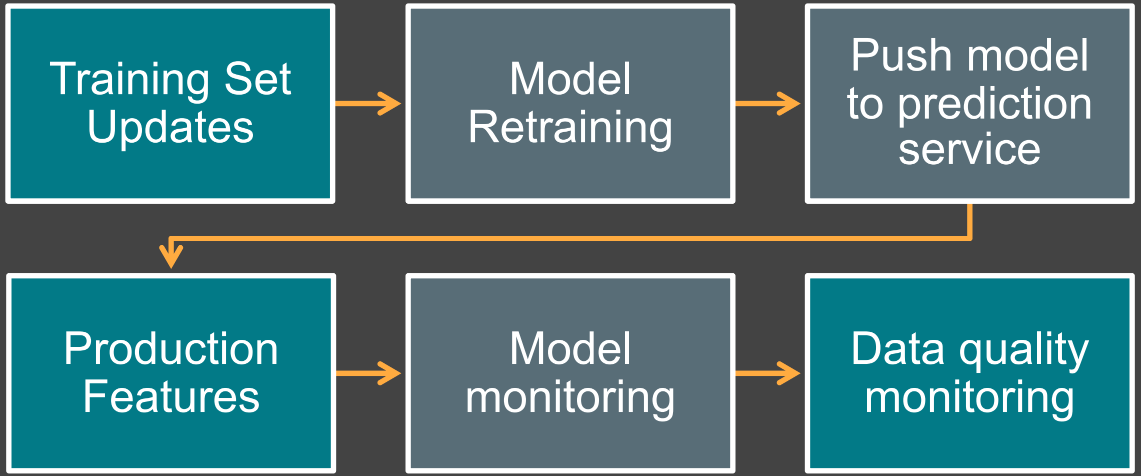
Zipline in the ML Iteration Workflow



Zipline in the ML Workflow

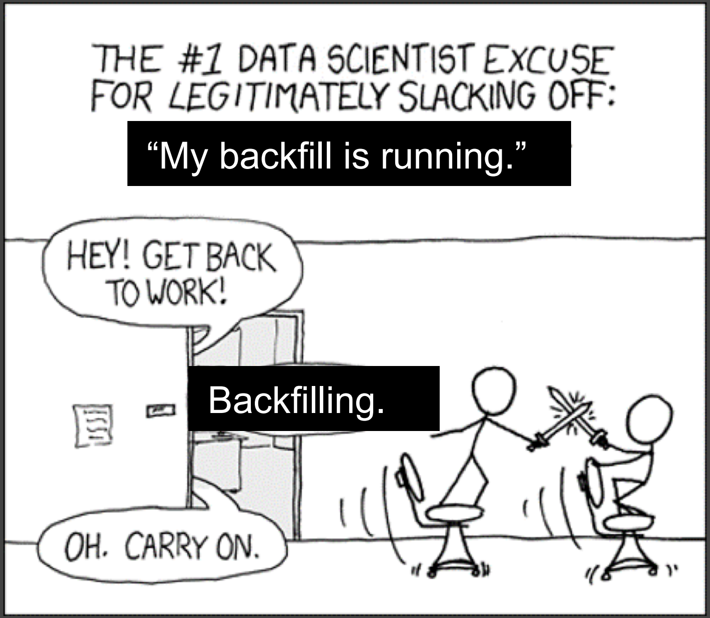


Zipline in the ML Production Workflow



Motivation

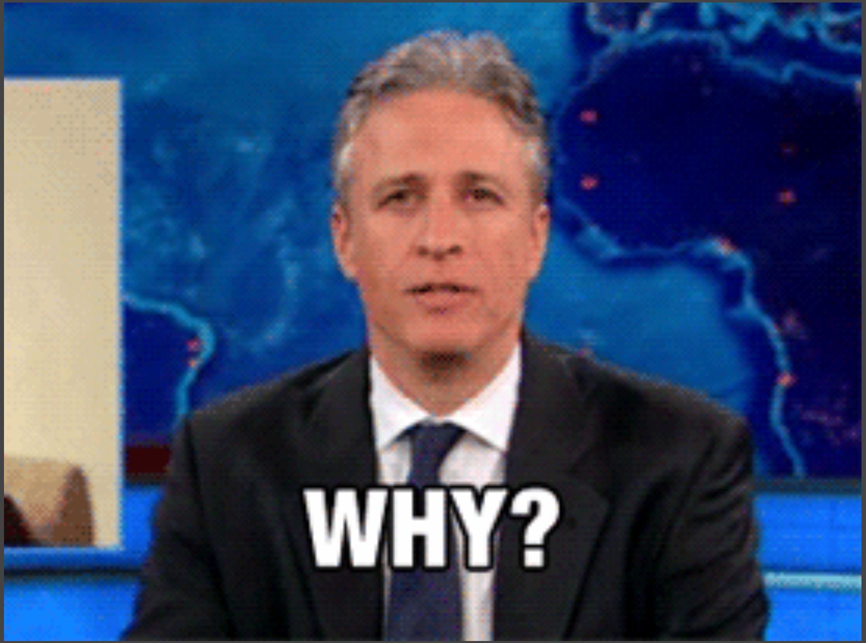
“I spend 60% of my time generating training data”



We already have a data warehouse

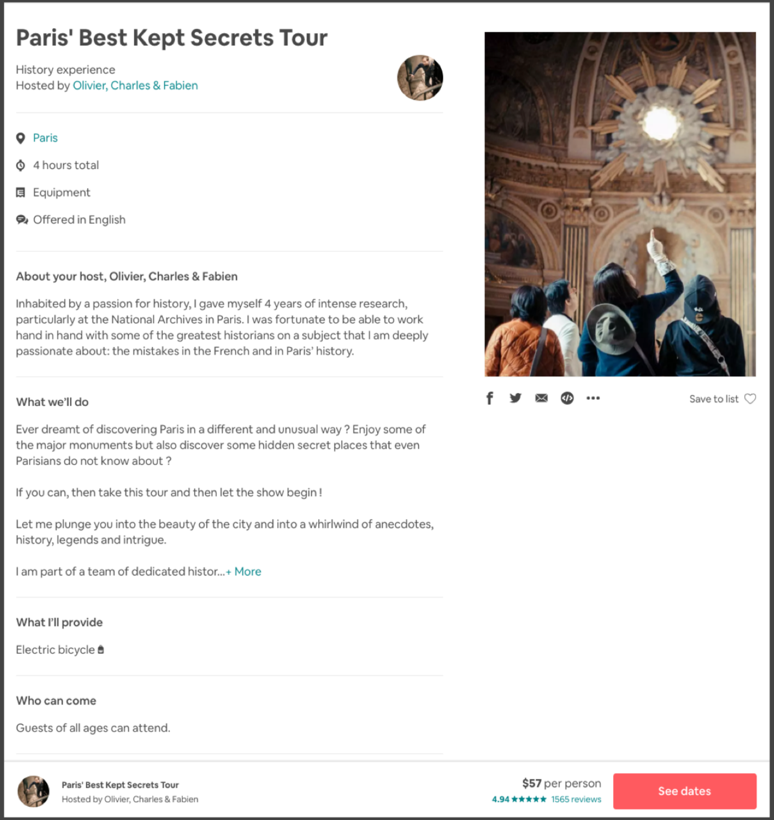
Why do we even need Zipline?

- We have data
- Defining new pipelines is easy enough (business analysts do it all the time)
- We already built a lot of tooling for all that
- Why build something new?

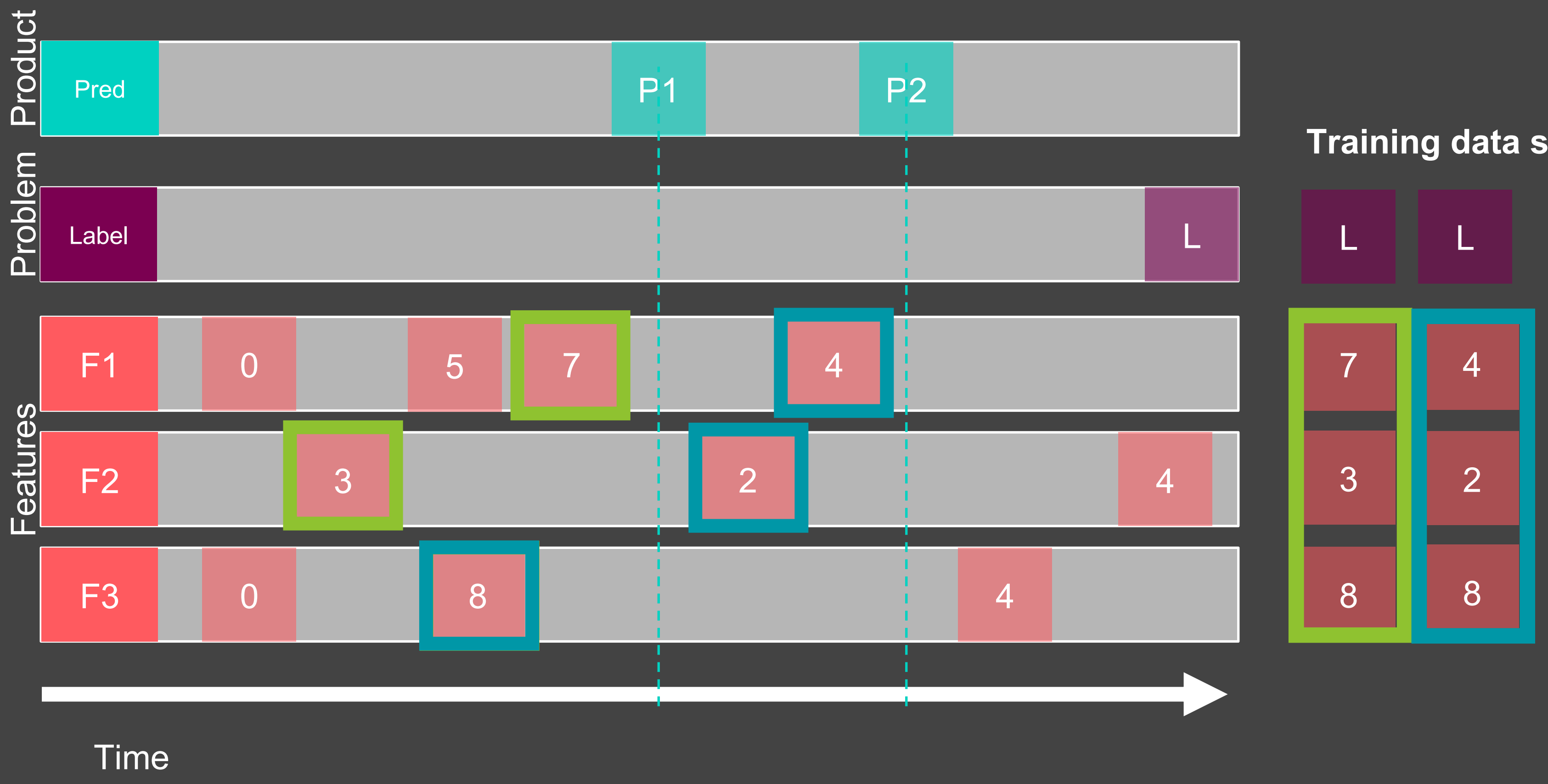


Motivating Example – Likelihood to book

- Predict likelihood to book when a user views an experience
- Example feature: sum of prior bookings in past 7 days



Timelines in ML Workflow



Limitations of a standard warehouse for Machine Learning: Bound to daily accuracy

Data warehouse (human consumption)

User	date	Sum of bookings
123	2018-01-01	1
123	2018-01-02	3

ML use case (machine consumption)

User	time	Sum of bookings
123	2018-01-01 11:15:24.142	**Is it 0 or 1?**
123	2018-01-02 18:15:24.142	**Is it 2 or 3?**

Limitations #1 of a standard warehouse for Machine Learning: Bound to daily accuracy

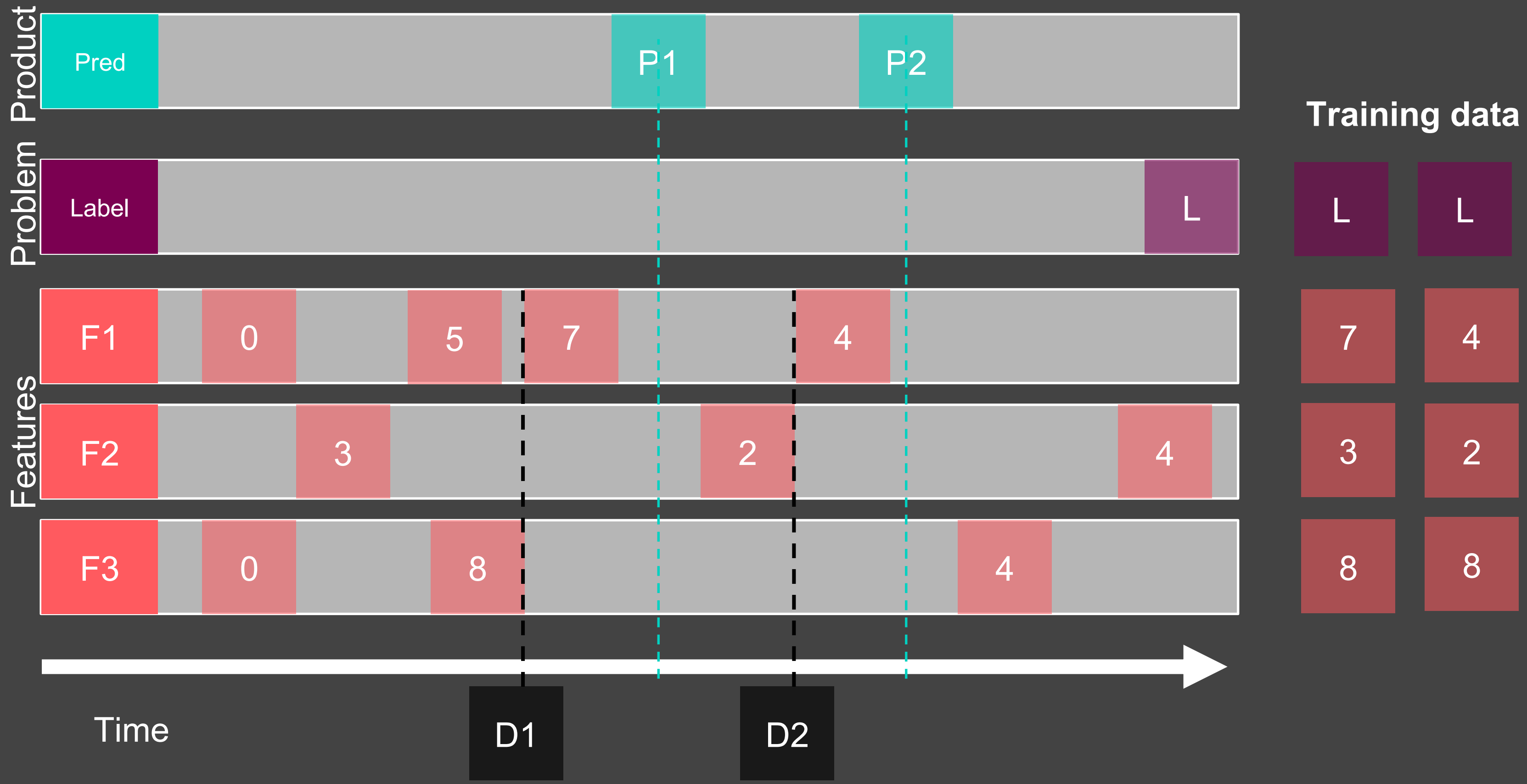
Data warehouse (human consumption)

User	date	Sum of bookings
123	2018-01-01	1
123	2018-01-02	3

ML use case (machine consumption)

User	time	Sum of bookings	Sum of bookings in past 12hrs
123	2018-01-01 11:15:24.142	**Is it 0 or 1?**	**???**
123	2018-01-02 18:15:24.142	**Is it 2 or 3?**	**???**

Timelines in ML Workflow



Why does this matter so much?

Just use the end of day value?

Why does this matter so much?

Just use the end of day value?

- Label leakage: **“My model performs well on the test data, but not in production. I don’t know how to debug.”**

Just use the start of day value?

Why does this matter so much?

Just use the end of day value?

- Label leakage: **“My model performs well on the test data, but not in production. I don’t know how to debug.”**

Just use the start of day value?

- You deprive your model of recent data
- Ex feature: number of searches in the past 24 hours.

If you missed that...

Point in time correctness is <u>important</u> and <u>hard</u>

We already have a production database

Why do we even need Zipline?

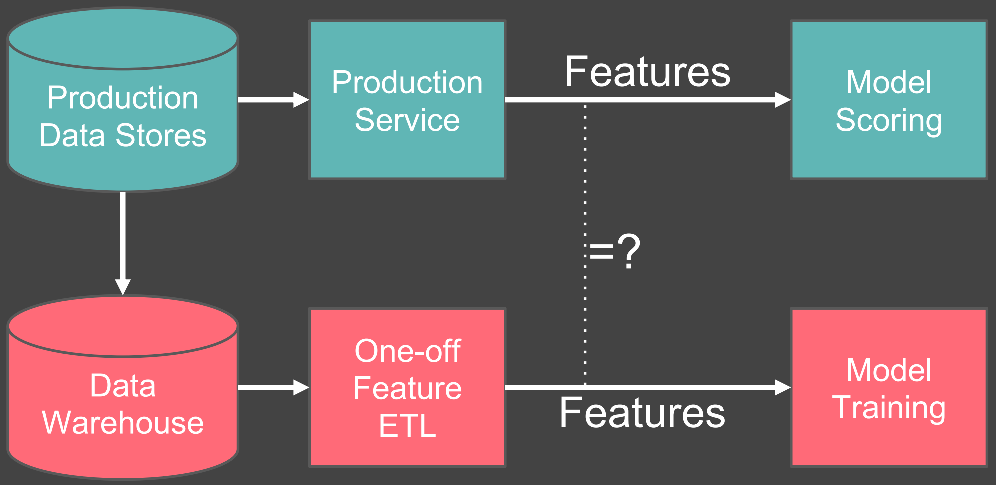
Production DB serves Airbnb.com just fine, surely it can handle “online” scoring traffic too?

1. Number of searches in the past 30 days? *Not in prod DB.*
2. Sum of bookings in past year. *airbnb.com goes down.*

We already have a production database

Why do we even need Zipline?

- We need the ***exact same data*** when ***training*** and ***scoring***



“My model performs well on the test data, but not in production. I don’t know how to debug.”

If you missed that again…

Point in time correctness

Consistent data across training/scoring

+ Data quality and monitoring

+ Sharing and discovery

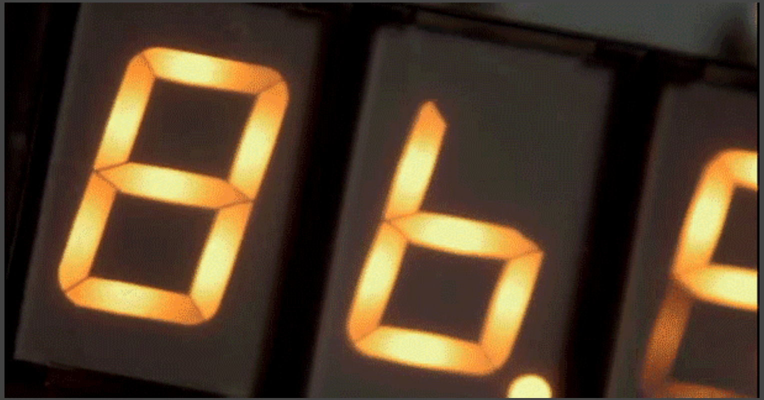
The Solution

Time Travel

Zipline puts a time machine on your data warehouse

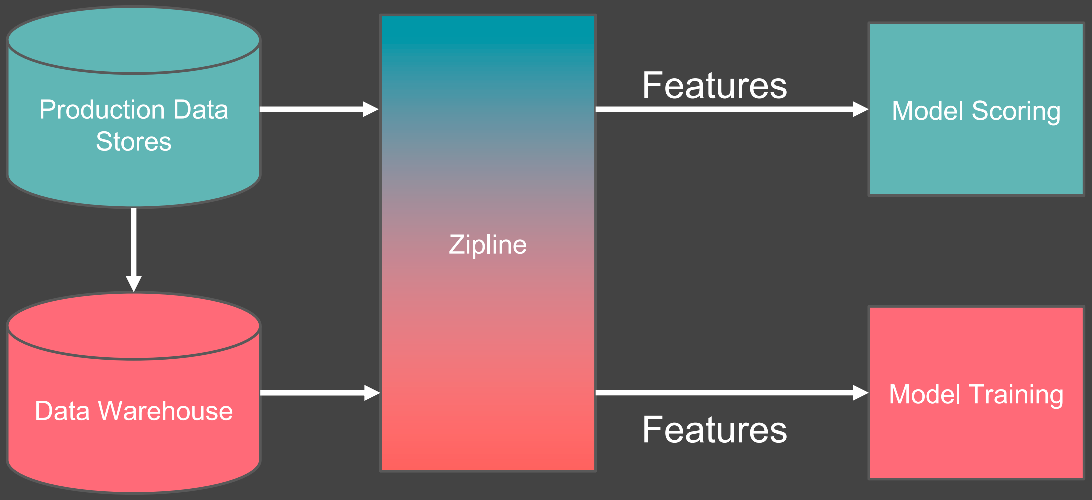
Your data warehouse

Your data warehouse with Zipline



Training/Predicting Consistency Guaranteed

Zipline travels through time *and space*



Other requirements

- Monitoring
- Sharing
- Integrate smoothly with the bigger picture ML workflow (see bighead)

User Interface

Feature Sharing and Discovery

1. Searchable
2. Easily understandable
3. Find outliers
4. Identify transformations
5. Shopping cart

Features Definition (Time Travel)

1. Count the bookings
2. Average their values
3. 7d, 14d, 30d, 180d, 1y exact windows

Features Definition (Time Travel)

You define features in a way that allows point in time correct computations

```
source: {
  type: hive
  query: """
    SELECT
    , guest as user
    , ts
    , value as value
      1 as count
    FROM bookings
    WHERE
      ds BETWEEN '{{ start_date }}' AND '{{ end_date }}'
  """
  dependencies: [bookings]
}
```

```
features: {
  count: {
    doc: "Total bookings."
    column: count
    operation: sum
    windows: ["7d", "30d", "180d", "1y"]
  },
  avg_price: {
    doc: "Average booking value."
    column: value
    operation: avg
    windows: ["7d", "30d", "180d", "1y"]
  }
```

Features Definition (Time Travel)

- Now Zipline knows ***how*** to time travel that feature… What happens next?
- ***Nothing!*** Until someone asks for a point in time computation (what is the value for this user at this time).
- ZiplineSource API

```
from zipline.training_set import TrainingSet

features = [
  "user_bookings_past_7d",
  "user_bookings_past_30d",
  "listing_booking_requests_past_7d",
  "listing_booking_requests_past_30d"]

query = '''
SELECT
  id_guest as user,
  id_listing as listing,
  ts as ts
FROM some_table
WHERE ds BETWEEN '{{ start_date }}' AND '{{ end_date }}'
'''


team = 'ml_infra'
name = 'source_test_3'
start_date = '2018-07-01'

training_set = TrainingSet(team, name, features, query, start_date)
```

Features Definition (Time Travel)

User provides this

user	listing	time
123	567	2018-01-01 12:23:23.123
234	678	2018-01-01 22:11:22.321
345	789	2018-01-02 01:45:55.891

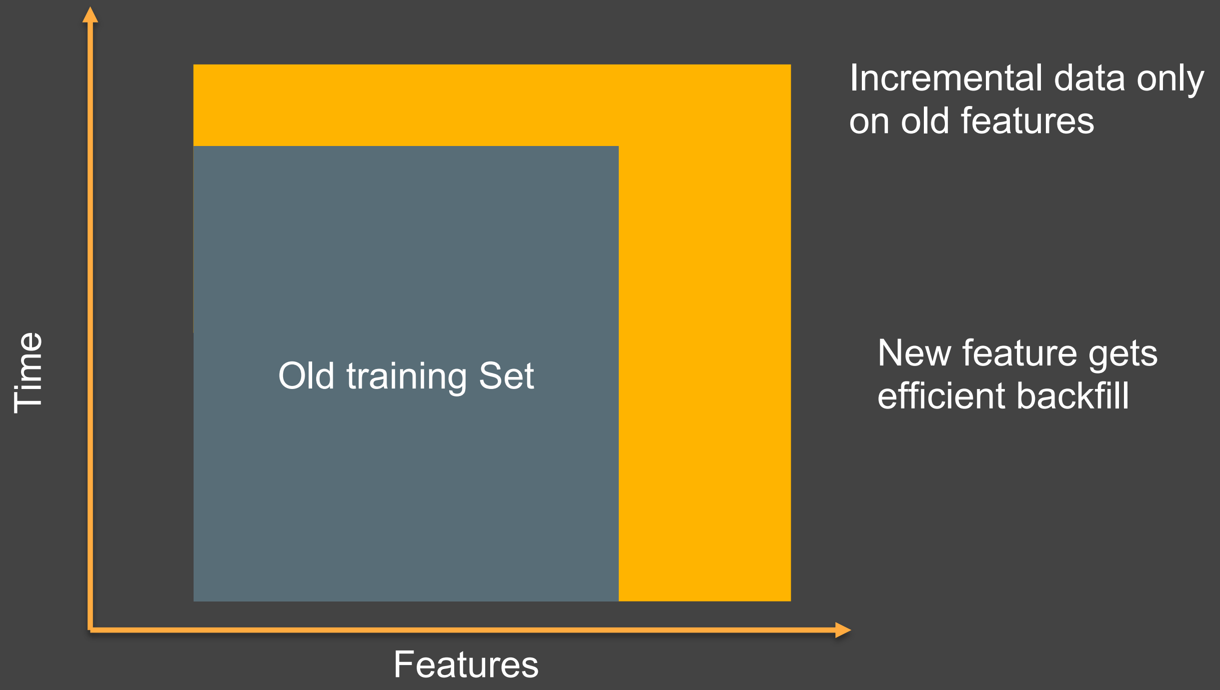
Features Definition (Time Travel)

User provides this | Zipline fills in this

user	listing	time	bookings_sum_7d	bookings_sum_14d
123	567	2018-01-01 ...	1	2
234	678	2018-01-01 …	4	4
345	789	2018-01-02 …	0	1

Features Definition (iteration)

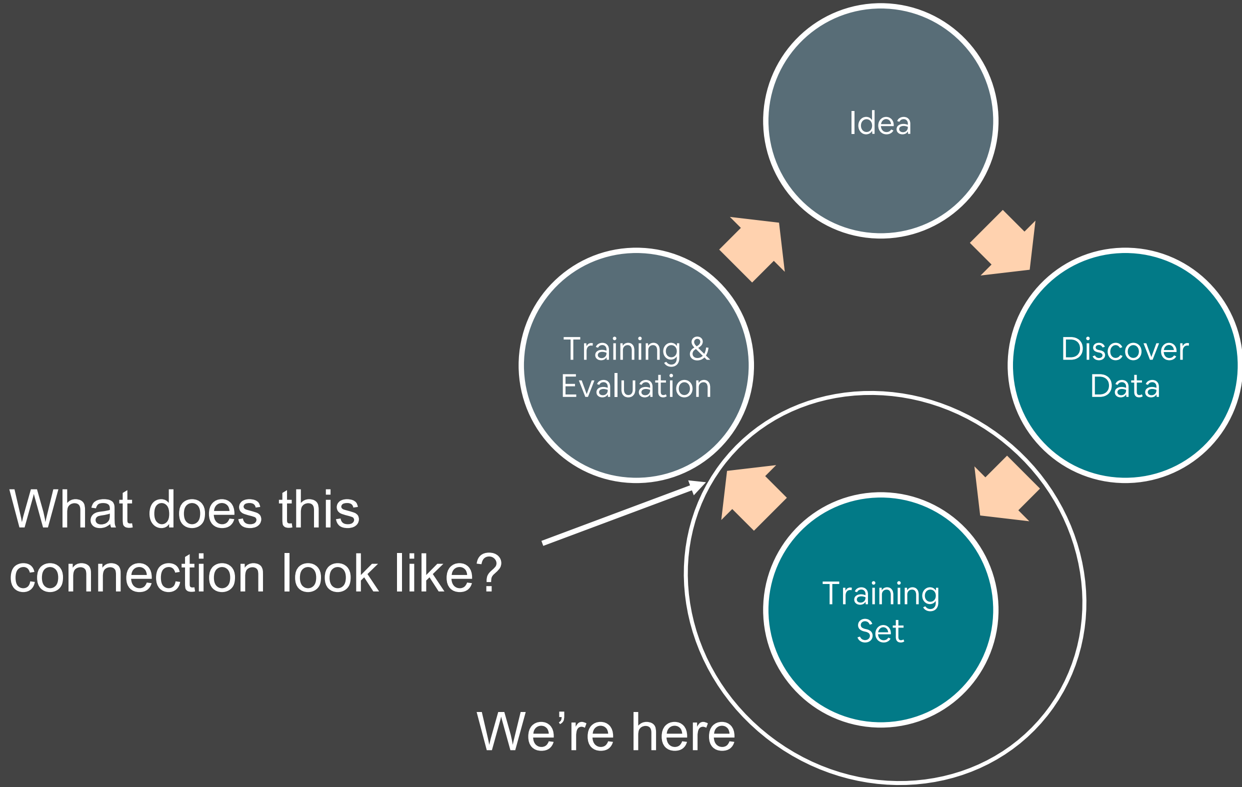
- Schema change? Don't worry about it
- Bugfixed a feature? There's an API for that



If you missed that again...

training set = f(features, primary keys, timestamps)

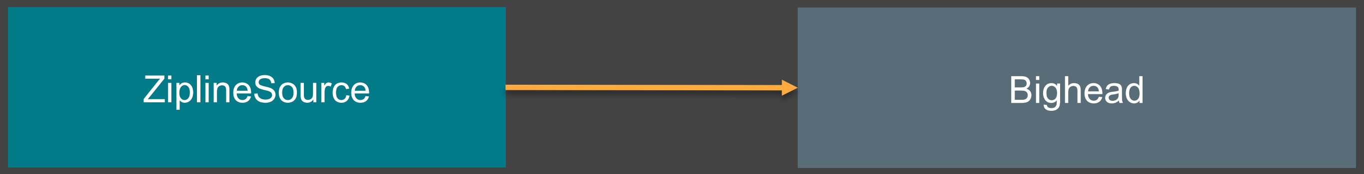
Zipline in the ML Iteration Workflow



Zipline in the ML Iteration Workflow

1. You build your Bighead model with a ZiplineSource
2. Configure it for daily training/scoring

Daily request to *get_{training/scoring}_dataframe()*



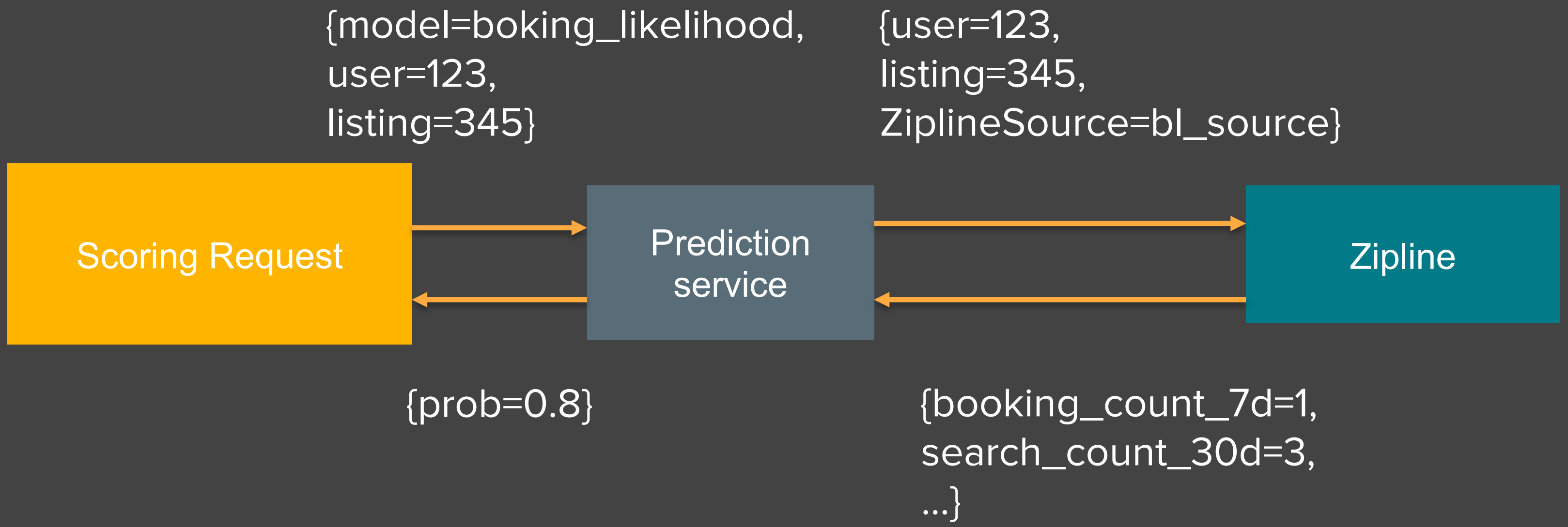
Zipline in the production workflow

- Bighead knows about your ZiplineSource
- ZiplineSource knows about your features

```
from zipline.training_set import TrainingSet

features = [
  "user_bookings_past_7d",
  "user_bookings_past_30d",
  "listing_booking_requests_past_7d",
  "listing_booking_requests_past_30d"]

query = '''
SELECT
  id_guest as user,
  id_listing as listing,
  ts as ts
FROM some_table
WHERE ds BETWEEN '{{ start_date }}' AND '{{ end_date }}'
'''


team = 'ml_infra'
name = 'source_test_3'
start_date = '2018-07-01'

training_set = TrainingSet(team, name, features, query, start_date)
```

Zipline in the production workflow

- Scoring requests only require primary key vectors (not feature vectors)



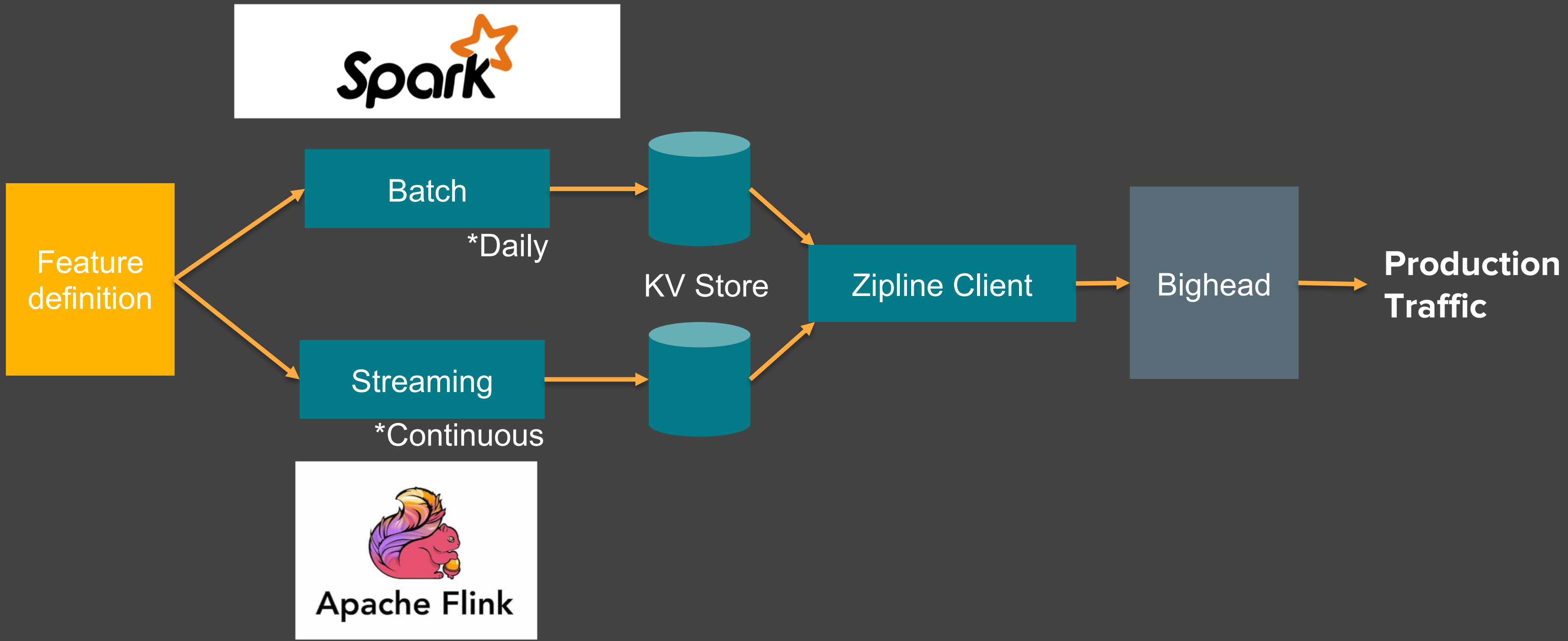
If you missed that again...

Features are the same in all environments

Further Technical Details

Train/Predict data consistency

Lambda Architecture



Time travel

How to do it efficiently

- Making this fast pays off
- Can get very expensive (many timestamps x many events)
- Skew is the enemy
- Caching partial aggregates can help
- Exact windows make it tricky

Time traveling on production DBs

Processing binlogs

- Daily dumps of production tables
- Lack of intra-day accuracy
- Zipline can ingest transaction logs
- Mutable events are tricky

Summary: Zipline is…

Time travel
Consistency
Data quality and monitoring
Searchable and sharable
Integrated with end-to-end workflow

Drumroll...

Open Sourcing Q1 2019

Reach out to andrew.hoh@airbnb.com for more info

Questions

Appendix

ZiplineSource API

ZiplineSource is a python API with two primary user facing functions

1. Get ***training*** dataframe (arguments for sampling, time ranges, etc.)
2. Get ***scoring*** dataframe (arguments for sampling, time ranges, etc.)