

Bighead

Airbnb's End-to-End Machine Learning Infrastructure

Andrew Hoh
ML Infra @ Airbnb



```
graph LR; A[Background] --> B[Design Goals]; B --> C[Architecture Deep Dive]; C --> D[Open Source]
```

Background

Design Goals

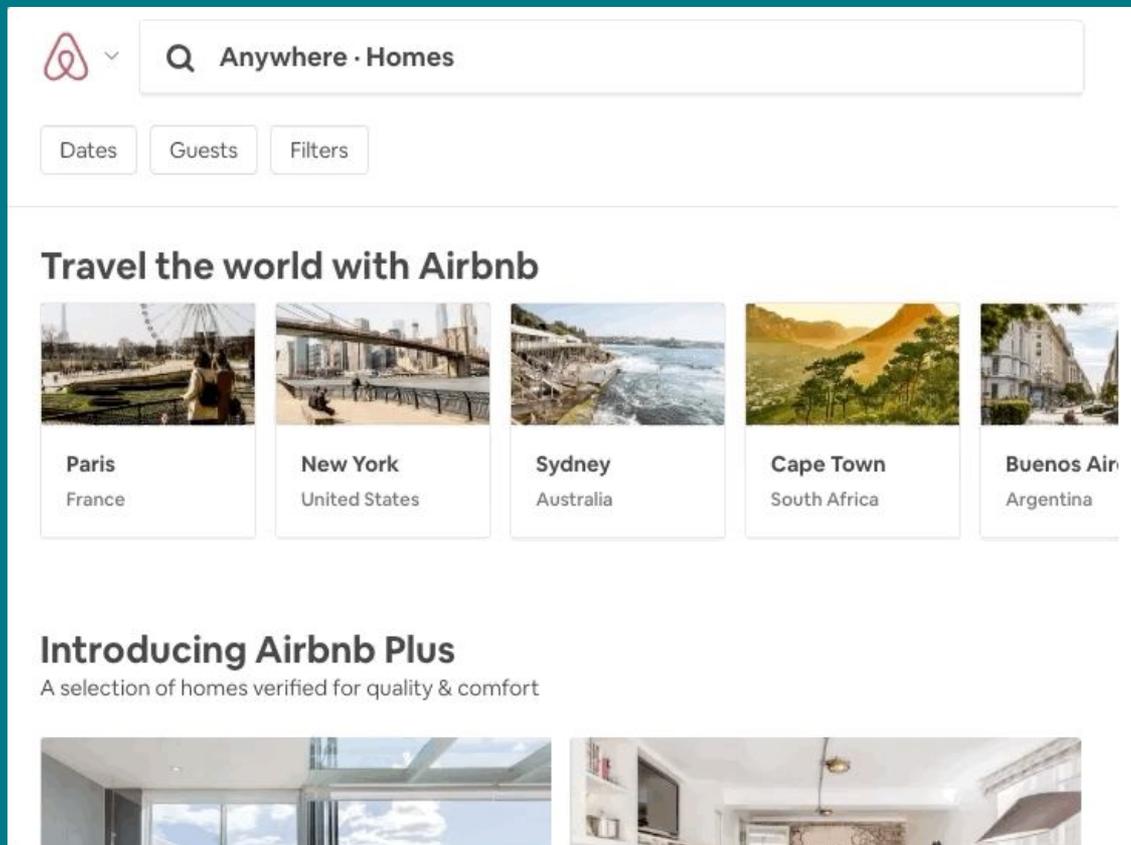
**Architecture
Deep Dive**

Open Source

Background

Airbnb's Product

A global travel community that offers magical end-to-end trips, including where you stay, what you do and the people you meet.



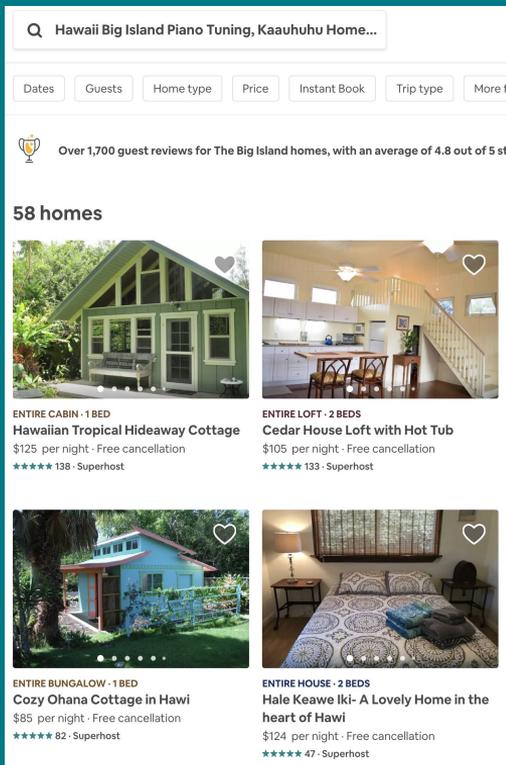
The screenshot displays the Airbnb homepage interface. At the top, there is a search bar with the Airbnb logo on the left and a search icon followed by the text "Anywhere · Homes". Below the search bar are three buttons: "Dates", "Guests", and "Filters".

The main content area features a section titled "Travel the world with Airbnb". This section contains five travel recommendation cards, each with a scenic image and text identifying the location and country:

- Paris**, France: Image of a park with a Ferris wheel in the background.
- New York**, United States: Image of a bridge over a city street.
- Sydney**, Australia: Image of a coastal city with waves.
- Cape Town**, South Africa: Image of a mountainous landscape with trees.
- Buenos Aires**, Argentina: Image of a city street with buildings.

Below this section is a heading "Introducing Airbnb Plus" followed by the subtext "A selection of homes verified for quality & comfort". At the bottom, there are two images showing the interior of a modern, bright home with large windows and a well-furnished living area.

Airbnb is already driven by Machine Learning



Q Hawaii Big Island Piano Tuning, Kaauihuhu Home...

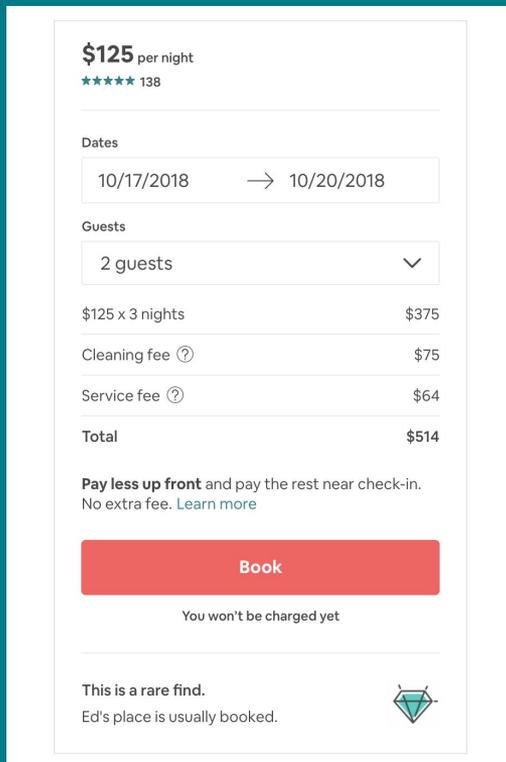
Dates Guests Home type Price Instant Book Trip type More filters

Over 1,700 guest reviews for The Big Island homes, with an average of 4.8 out of 5 stars

58 homes

- ENTIRE CABIN - 1 BED**
Hawaiian Tropical Hideaway Cottage
\$125 per night · Free cancellation
★★★★★ 138 · Superhost
- ENTIRE LOFT - 2 BEDS**
Cedar House Loft with Hot Tub
\$105 per night · Free cancellation
★★★★★ 133 · Superhost
- ENTIRE BUNGALOW - 1 BED**
Cozy Ohana Cottage in Hawi
\$85 per night · Free cancellation
★★★★★ 82 · Superhost
- ENTIRE HOUSE - 2 BEDS**
Hale Keawe Iki - A Lovely Home in the heart of Hawi
\$124 per night · Free cancellation
★★★★★ 47 · Superhost

Search Ranking



\$125 per night
★★★★★ 138

Dates
10/17/2018 → 10/20/2018

Guests
2 guests

\$125 x 3 nights	\$375
Cleaning fee ?	\$75
Service fee ?	\$64
Total	\$514

Pay less up front and pay the rest near check-in. No extra fee. [Learn more](#)

Book

You won't be charged yet

This is a rare find. Ed's place is usually booked.

Smart Pricing



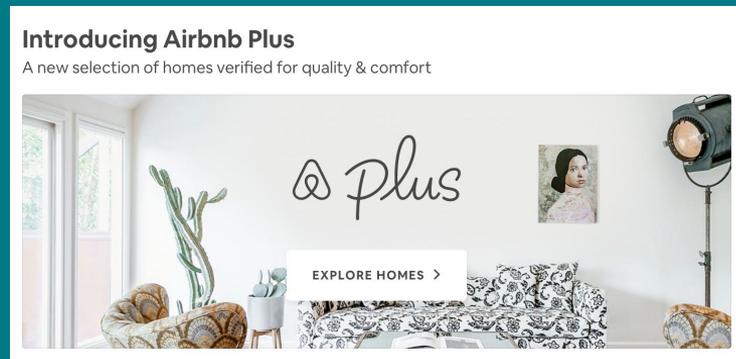
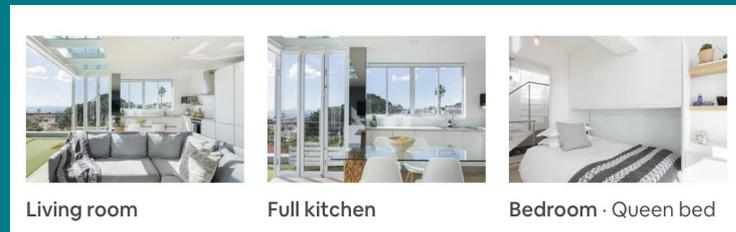
Risk scoring

Every Airbnb reservation is scored for risk before it's confirmed. We use predictive analytics and machine learning to instantly evaluate hundreds of signals that help us flag and investigate suspicious activity before it happens.

Fraud Detection

But there are **many** more opportunities for ML

- Paid Growth - Hosts
- Classifying / Categorizing Listings
- Experience Ranking + Personalization
- Room Type Categorizations
- Customer Service Ticket Routing
- Airbnb Plus
- Listing Photo Quality
- Object Detection - Amenities
-



Intrinsic Complexities with Machine Learning

- Understanding the business domain
- Selecting the appropriate Model
- Selecting the appropriate Features
- Fine tuning

Incidental Complexities with Machine Learning

- Integrating with Airbnb's Data Warehouse
- Scaling model training & serving
- Keeping consistency between: Prototyping vs Production, Training vs Inference
- Keeping track of multiple models, versions, experiments
- Supporting iteration on ML models

→ ML models take on average 8 to 12 weeks to build

→ ML workflows tended to be slow, fragmented, and brittle

The ML Infrastructure Team addresses these challenges

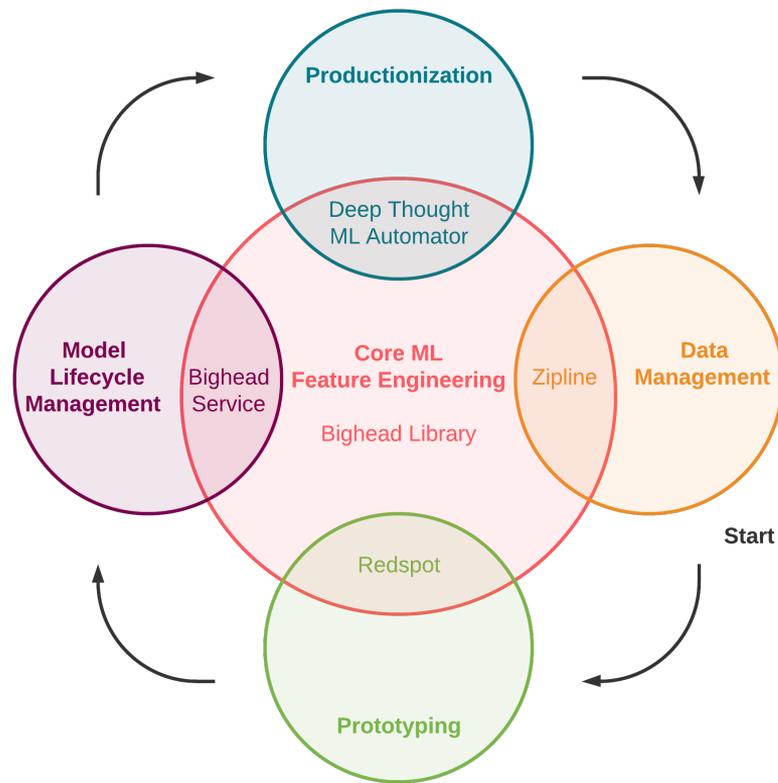
Vision

Airbnb routinely ships ML-powered features throughout the product.

Mission

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

Supporting the Full ML Lifecycle



Bighead: Design Goals

Seamless

Versatile

Consistent

Scalable

Seamless

- Easy to prototype, easy to productionize
- Same workflow across different frameworks

Versatile

- Supports all major ML frameworks
- Meets various requirements
 - Online and Offline
 - Data size
 - SLA
 - GPU training
 - Scheduled and Ad hoc

Consistent

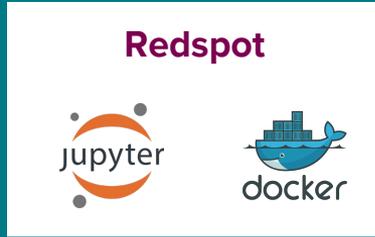
- Consistent environment across the stack
- Consistent data transformation
 - Prototyping and Production
 - Online and Offline

Scalable

- Horizontal
- Elastic

Bighead: Architecture Deep Dive

Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

Prototyping

Lifecycle Management

Production

Redspot



jupyter docker



Bighead Service / UI

Deep Thought



kubernetes

Real Time Inference

ML Automator



Airflow
APACHE SPARK

Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

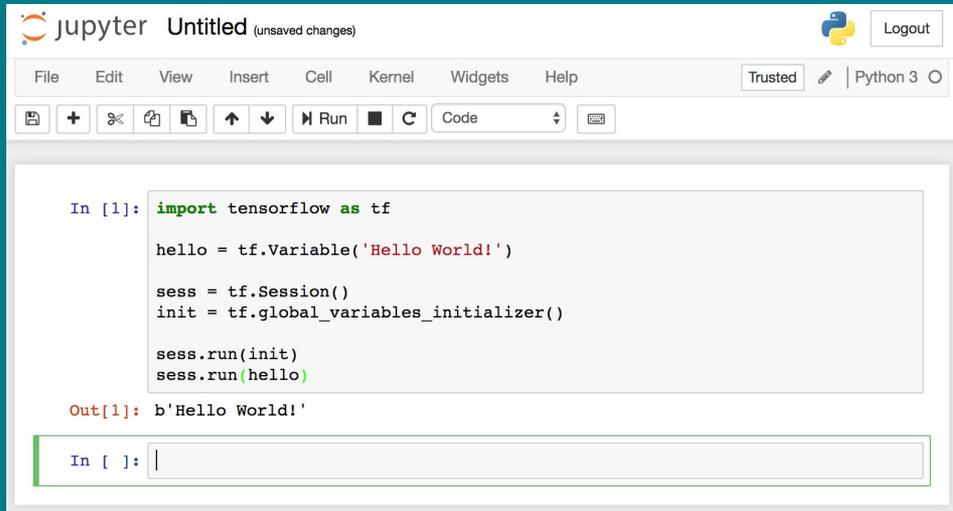
Feature Data Management: **Zipline**

Redspot

Prototyping with Jupyter Notebooks

Jupyter Notebooks?

What are those?



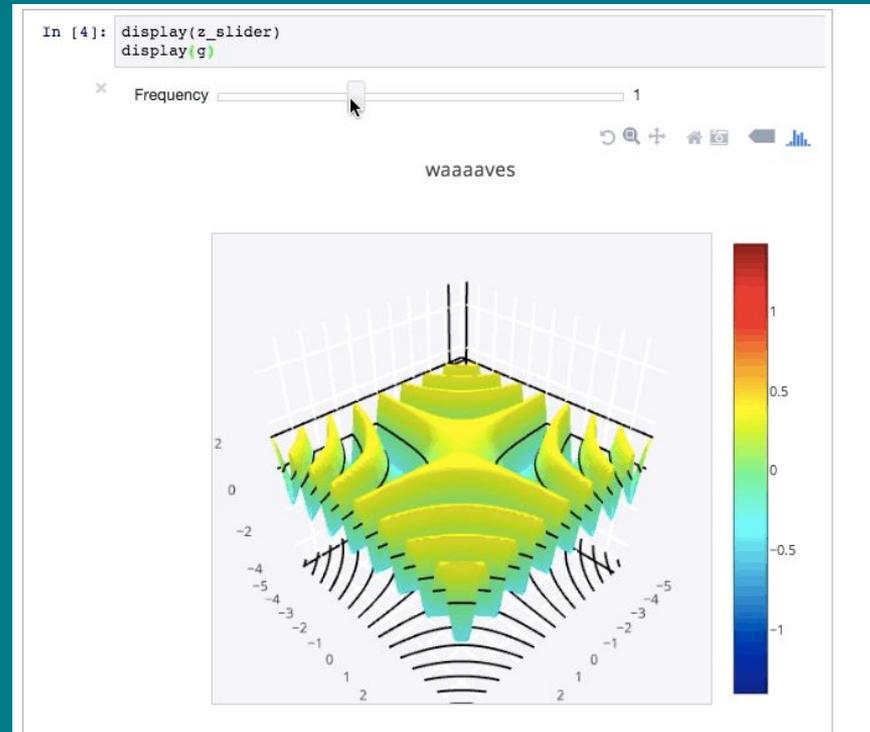
The screenshot shows the Jupyter Notebook interface. The top bar includes the Jupyter logo, the text "jupyter Untitled (unsaved changes)", and a "Logout" button. Below the top bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A "Trusted" status indicator and "Python 3" are also visible. The main area contains a code cell with the following Python code:

```
In [1]: import tensorflow as tf
hello = tf.Variable('Hello World!')
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
sess.run(hello)
```

The output of the cell is:

```
Out[1]: b'Hello World!'
```

Below the output is an empty input cell for the next command.



The screenshot shows the Jupyter Notebook interface with a code cell containing:

```
In [4]: display(z_slider)
display(g)
```

Below the code cell is a "Frequency" slider widget with a value of 1. Below the slider is the text "waaaaves". Below the text is a 3D surface plot of a function. The plot shows a complex, multi-peaked surface with a color scale on the right ranging from -1 (blue) to 1 (red). The axes are labeled with values from -5 to 2.

“Creators need an immediate connection
to what they are creating.”
- Bret Victor

The ideal Machine Learning development environment?

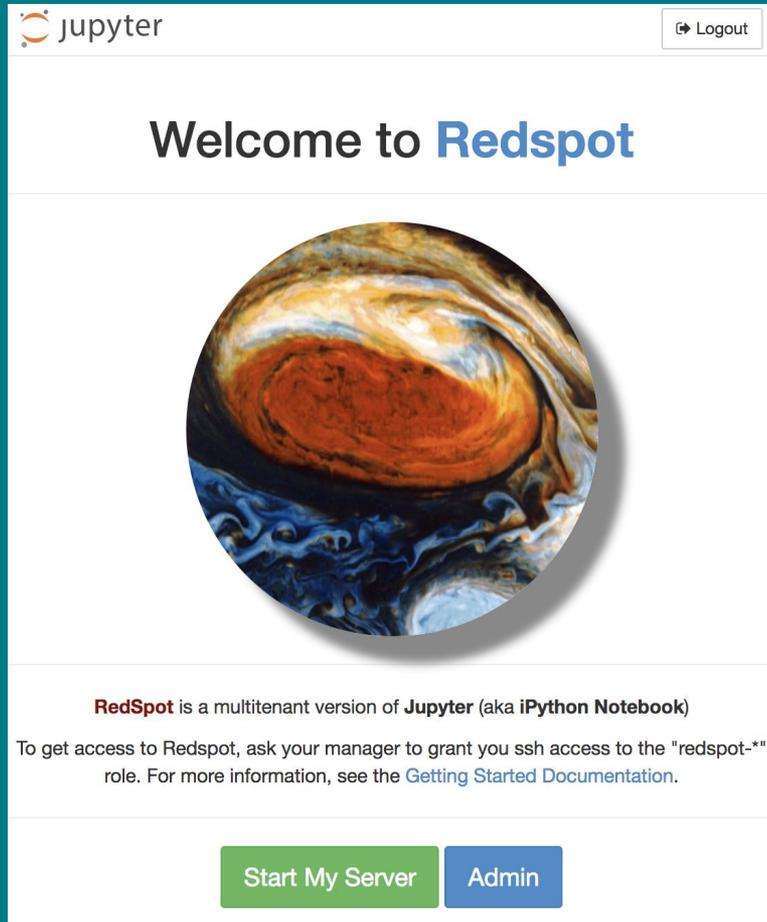
- Interactivity and Feedback
- Access to Powerful Hardware
- Access to Data



Redspot

a Supercharged Jupyter Notebook Service

- A fork of the JupyterHub project
- Integrated with our Data Warehouse
- Access to specialized hardware (e.g. GPUs)
- File sharing between users via AWS EFS
- Packaged in a familiar Jupyterhub UI



The screenshot shows the Redspot landing page. At the top left is the Jupyter logo and the word "jupyter". At the top right is a "Logout" button. The main heading is "Welcome to Redspot". Below this is a large circular image of a Jupiter storm. Underneath the image, it says "RedSpot is a multitenant version of Jupyter (aka iPython Notebook)". Below that is a paragraph: "To get access to Redspot, ask your manager to grant you ssh access to the 'redspot-*' role. For more information, see the [Getting Started Documentation](#)." At the bottom are two buttons: "Start My Server" and "Admin".

jupyter Logout

Welcome to Redspot



RedSpot is a multitenant version of **Jupyter** (aka **iPython Notebook**)

To get access to Redspot, ask your manager to grant you ssh access to the "redspot-*" role. For more information, see the [Getting Started Documentation](#).

[Start My Server](#) [Admin](#)

Redspot

Logout

Choose your Jupyter environment

Select a job profile:

Remote Docker

Docker image configuration:

Ubuntu 18.04 image with python 2.7 (for CPU)

Instance configuration Edit

Instance Type: t2.medium
Billing Group: ml_infra

Launch

Instance Config ✕

Launch a new instance

- t2.medium (minimal environment)
- c4.xlarge (CPU-optimized (4 CPUs))
- r4.2xlarge (memory-optimized 61 GB)
- p2.xlarge (Nvidia Tesla K80 x 1)

Redspot

a Supercharged Jupyter Notebook Service

Consistent

- Promotes prototyping in the exact environment that your model will use in production

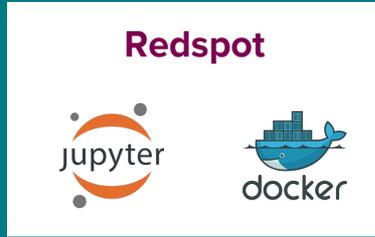
Versatile

- Customized Hardware: AWS EC2 Instance Types e.g. P3, X1
- Customized Dependencies: Docker Images e.g. Py2.7, Py3.6+Tensorflow

Seamless

- Integrated with Bighead Service & Docker Image Service via APIs & UI widgets

Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

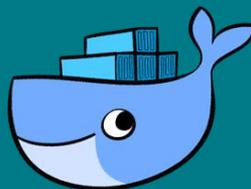
Feature Data Management: **Zipline**

Docker Image Service

Environment Customization

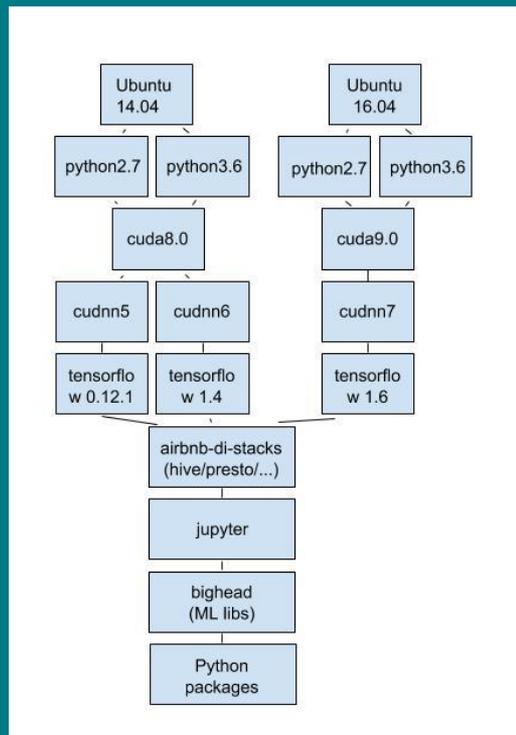
Docker Image Service - Why

- ML Users have a diverse, heterogeneous set of dependencies
- Need an easy way to bootstrap their own runtime environments
- Need to be consistent with the rest of Airbnb's infrastructure

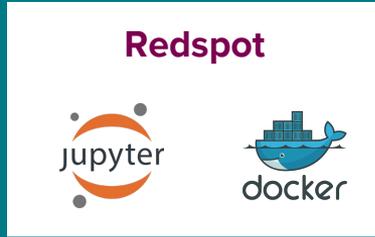


Docker Image Service - Dependency Customization

- Our configuration management solution
- A composition layer on top of Docker
- Includes a customization service that faces our users
- Promotes **Consistency** and **Versatility**



Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

Bighead Service

Model Lifecycle Management

Model Lifecycle Management - why?

- Tracking ML model changes is just as important as tracking code changes
- ML model work needs to be **reproducible** to be **sustainable**
- Comparing experiments before you launch models into production is critical



Introducing Bighead

Explore, deploy, and monitor machine learning models.

[learn more](#)

Showing 15 machine learning projects

Name	Owner	Active Artifact Name
census_income_xgb_classification	conglei_shi	auto_trained_model #0
price_prediction_keras	conglei_shi	auto_trained_model #0-0
census_income_pytorch_classification	conglei_shi	auto_trained_model #18-1
image_classification_tensorflow	conglei_shi	auto_trained_model #0-2
image_classification_pytorch_CNN	conglei_shi	N/A
speech_recognition_spanish	conglei_shi	auto_trained_model #0-19
census_income_mxnet_classification	conglei_shi	auto_trained_model #0-20
speech_recognition_english	conglei_shi	auto_trained_model #0-21
translation_model_eng_esp	conglei_shi	auto_trained_model #0-22
object_detection_image_tensorflow	conglei_shi	auto_trained_model #0-23
vr_listing_spark	conglei_shi	auto_trained_model #0-24
translation_model_eng_chn	conglei_shi	auto_trained_model #0-25

Cluster

- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace

bighead-online-models ▾**Overview****Workloads**

Cron Jobs

CPU usage**Memory usage** ⓘ**Pods**

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
------	------	--------	----------	-----	-------------	----------------

Project: census_income_xgb_classification

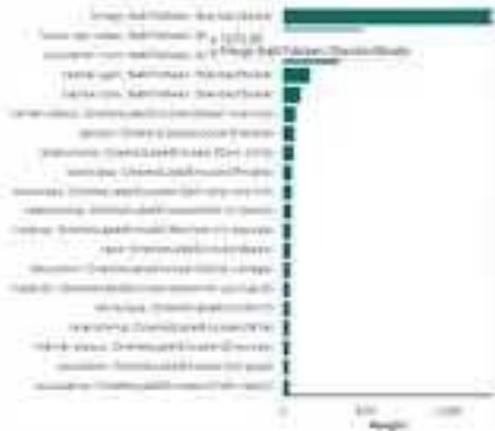
Details
 Active Space Model
 with Census Income Model
 Version of 2018
 2017 Task

Training Workflow
 Starting from 2018 09-11
 at 12:00 AM

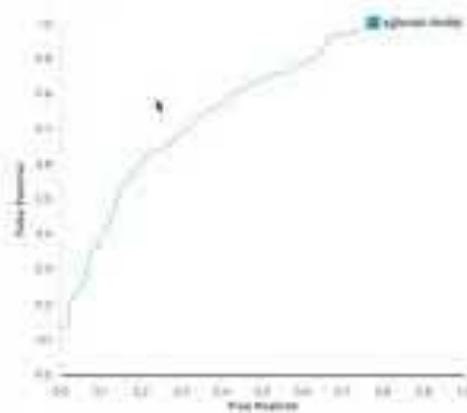
Score
 0.91416

Performance **Plots**

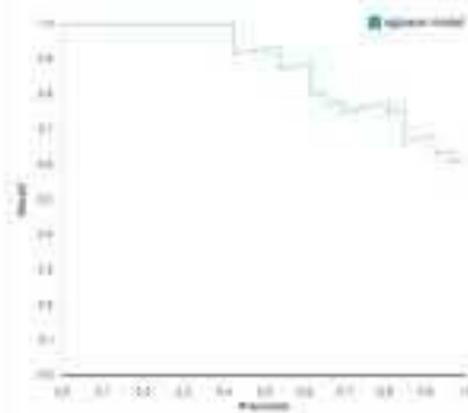
Feature Importance



ROC Curve



Precision-Recall Curve



Bighead Service

Consistent

- Central model management service
- Single source of truth about the state of a model, it's dependencies, and what's deployed

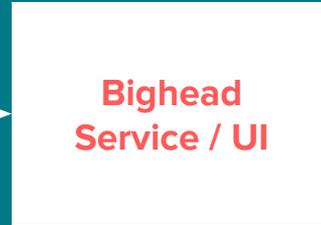
Seamless

- Context-aware visualizations that carry over from the prototyping experience

Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

Bighead Library

ML Models are highly heterogeneous in

Frameworks



Training data

- Data quality
- Structured vs Unstructured (image, text)

Environment

- GPU vs CPU
- Dependencies

ML Models are hard to keep consistent

- Data in *production* is different from data in *training*
- *Offline* pipeline is different from *online* pipeline
- Everyone does everything in a *different* way

Bighead Library

Versatile

- Pipeline on steroids - compute graph for preprocessing / inference / training / evaluation / visualization
- Composable, Reusable, Shareable
- Support popular frameworks



- Fast primitives for preprocessing
- Metadata for trained models

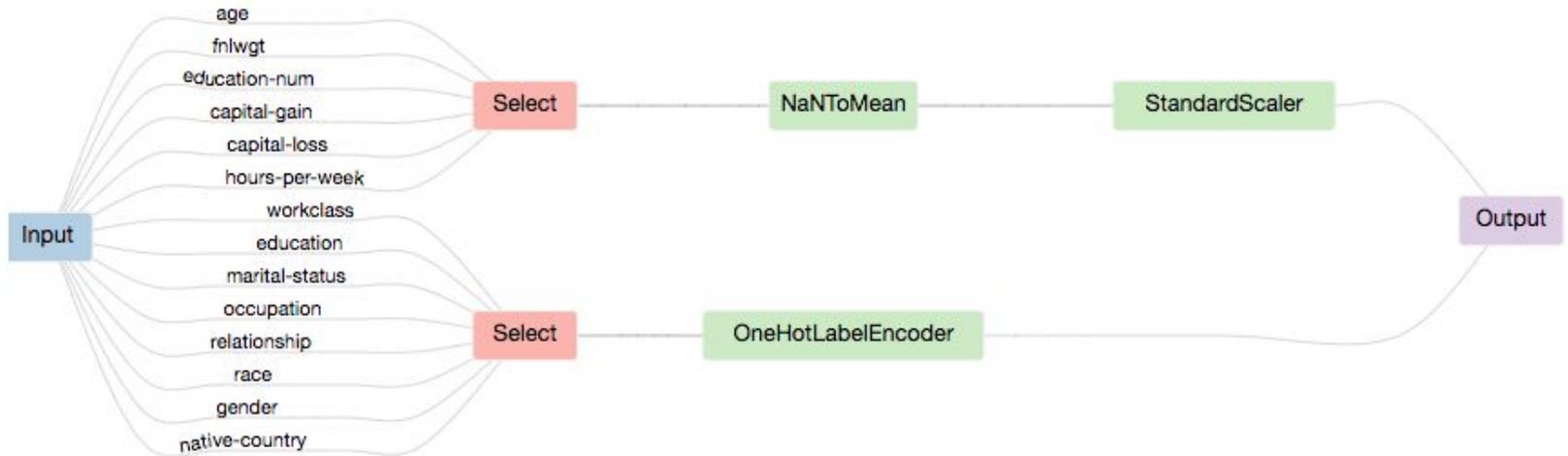
Consistent

- Uniform API
- Serializable - same pipeline used in training, offline inference, online inference

Bighead Library: ML Pipeline

```
categorical = [  
    'workclass',  
    'education',  
    'marital-status',  
    'occupation',  
    'relationship',  
    'race',  
    'gender',  
    'native-country',  
]  
  
numeric = [  
    'age',  
    'fnlwt',  
    'education-num',  
    'capital-gain',  
    'capital-loss',  
    'hours-per-week',  
]  
  
p = Pipeline('ClassifyCensusIncome')  
p[numeric] >>= [NaNToMean(dtype=np.float32), StandardScaler()]  
p[categorical] >>= OneHotLabelEncoder()  
p >>= XGBClassifier(objective='binary:logistic',  
                    n_estimators=100,  
                    learning_rate=0.1,  
                    max_depth=5)
```

Visualization - Pipeline



Easy to Serialize/Deserialize

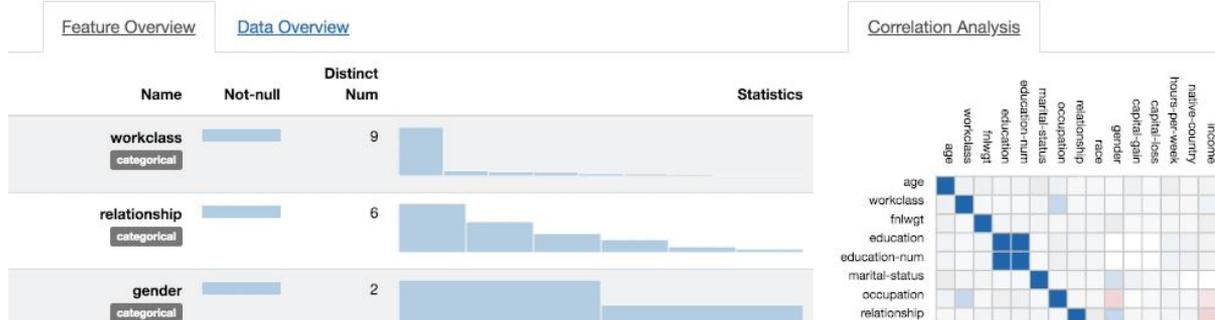
```
In [ ]: p.serialize('test.tar.xz')
```

```
In [ ]: p2 = Pipeline.deserialize('test.tar.xz')
```

Visualization - Training Data

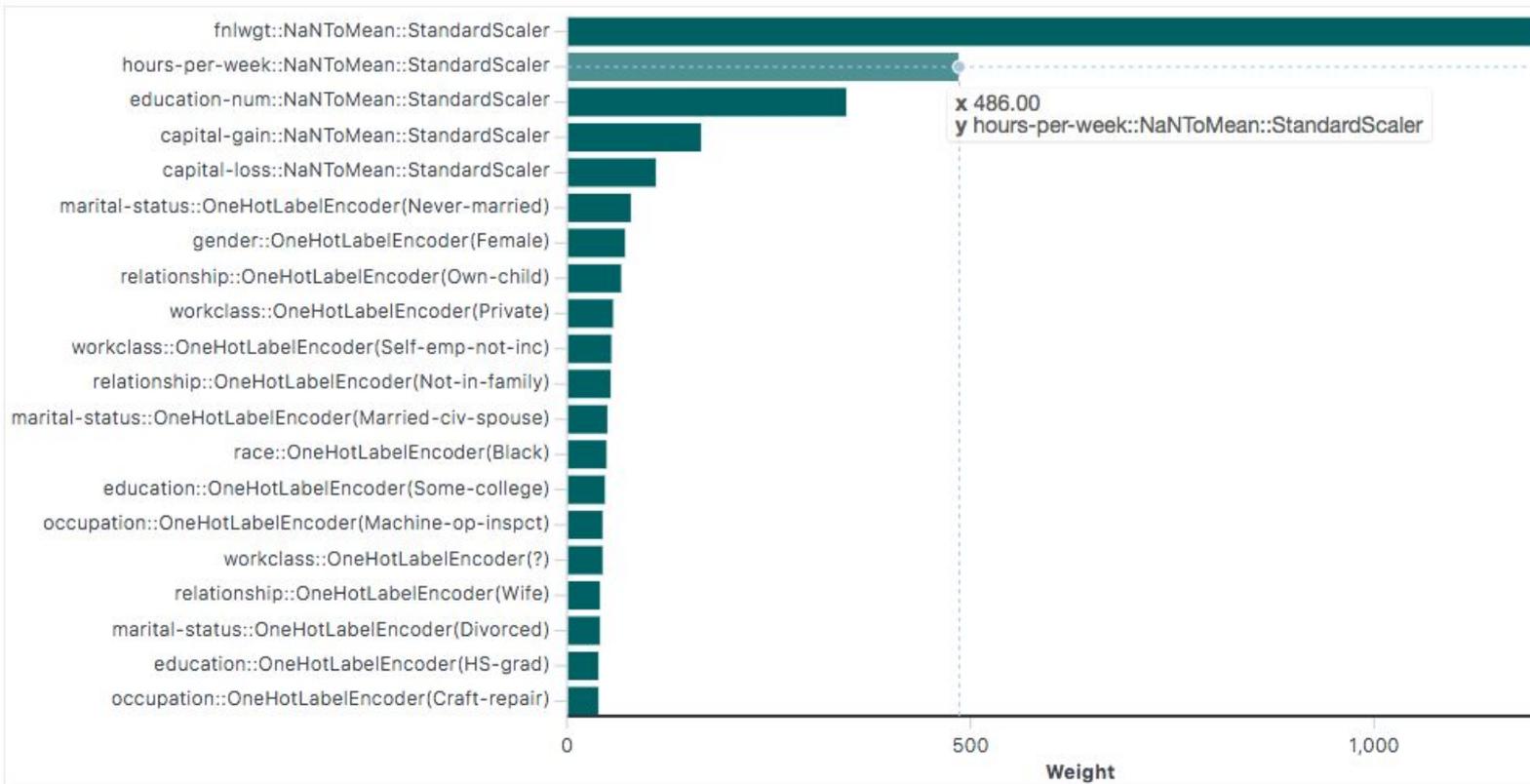
```
In [2]: train, test = load_census_income()
categorical = [
    'workclass',
    'education',
    'marital-status',
    'occupation',
    'relationship',
    'race',
    'gender',
    'native-country',
]
numeric = [
    'age',
    'fnlwgt',
    'education-num',
    'capital-gain',
    'capital-loss',
    'hours-per-week',
]
labels = np.array([0 if x== '<=50K' else 1 for x in train['income'].values])
```

```
In [3]: from bighead.core.visualization import show_stats
show_stats(train)
```

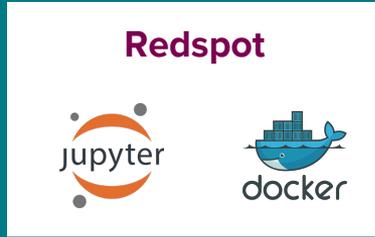


Visualization - Transformer

```
In [8]: from bighead.core.visualization import visualize
visualize(xgboost.get_feature_importance_metadata()[0])
```



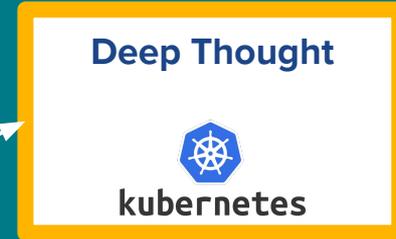
Prototyping



Lifecycle Management



Production



Real Time
Inference



Batch
Training +
Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

Deep Thought

Online Inference

Hard to make online model serving...

Consistent with training

- Different data
- Different pipeline
- Different dependencies

Easy to do

- Data scientists can't launch models without engineer team
- Engineers often need to rebuild models

Scalable

- Resource requirements varies across models
- Throughput fluctuates across time

Deep Thought

Consistent

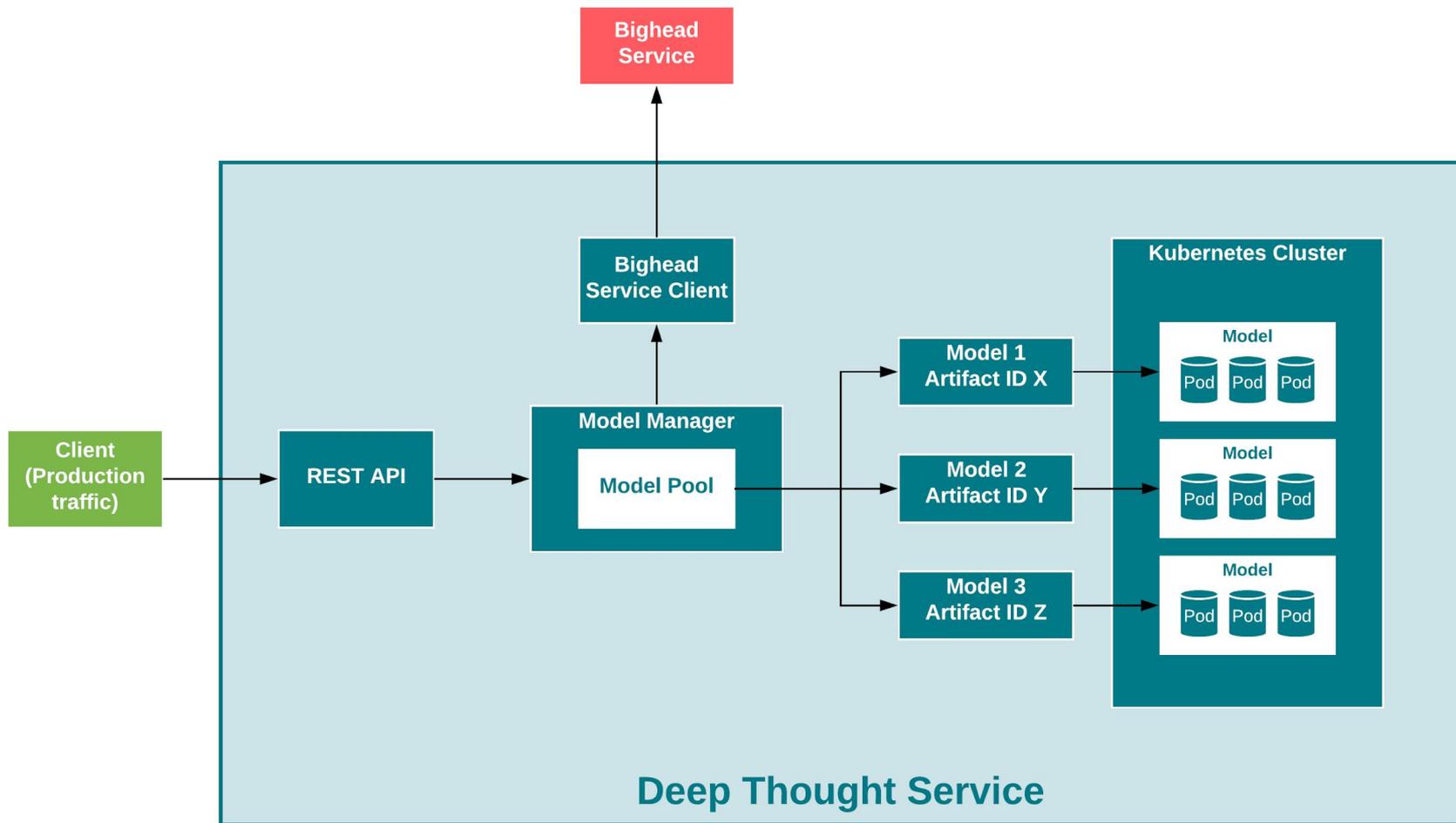
- Docker + Bighead Library: Same data source, pipeline, environment from training

Seamless

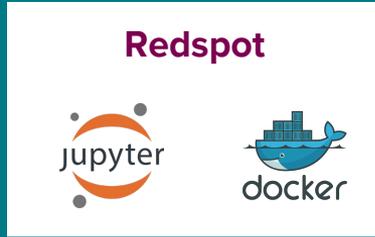
- Integration with event logging, dashboard
- Integration with Zipline

Scalable

- Kubernetes: Model pods can easily scale
- Resource segregation across models



Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

ML Automator

Offline Training and Batch Inference

ML Automator - Why

Automated training, inference, and evaluation are necessary

- Scheduling
- Resource allocation
- Saving results
- Dashboards and alerts
- Orchestration

ML Automator

Consistent

- Docker + Bighead Library: Same data source, pipeline, environment across the stack

Seamless

- Automate tasks via Airflow: Generate DAGs for training, inference, etc. with appropriate resources
- Integration with Zipline for training and scoring data

Scalable

- Spark: Distributed computing for large datasets

ML Automator

On DAG: bighead__vacation_rental_listing_spark_alfredo_luque_fit

schedule: 0 0 * * *

Graph View

Tree View

Task Duration

Task Tries

Landing Times

Gantt

Details

Code

Refresh

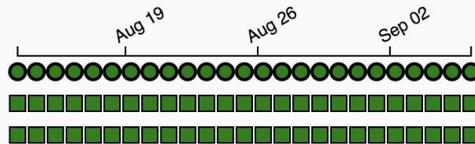
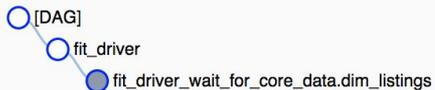
Base date: 2018-09-06 00:00:00

Number of runs: 25

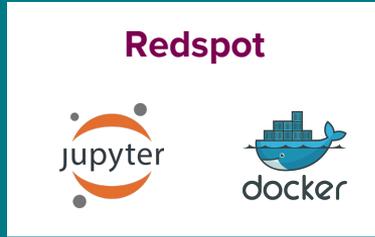
Go

BigheadFitOperator NamedHivePartitionSensor

success running failed skipped retry queued no status



Prototyping



Lifecycle Management



Production



Real Time Inference



Batch Training + Inference

Environment Management: **Docker Image Service**

Execution Management: **Bighead Library**

Feature Data Management: **Zipline**

Zipline

ML Data Management Framework

Feature management is hard

- **Inconsistent** offline and online datasets
- **Tricky** to generate training sets that depend on time correctly
- **Slow** training sets backfill
- **Inadequate** data quality checks or monitoring
- **Unclear** feature ownership and sharing

Zipline

Consistent

- Consistent data across training/scoring
- Consistent data across development/production
- Point-in-time correctness across features to prevent label leakage

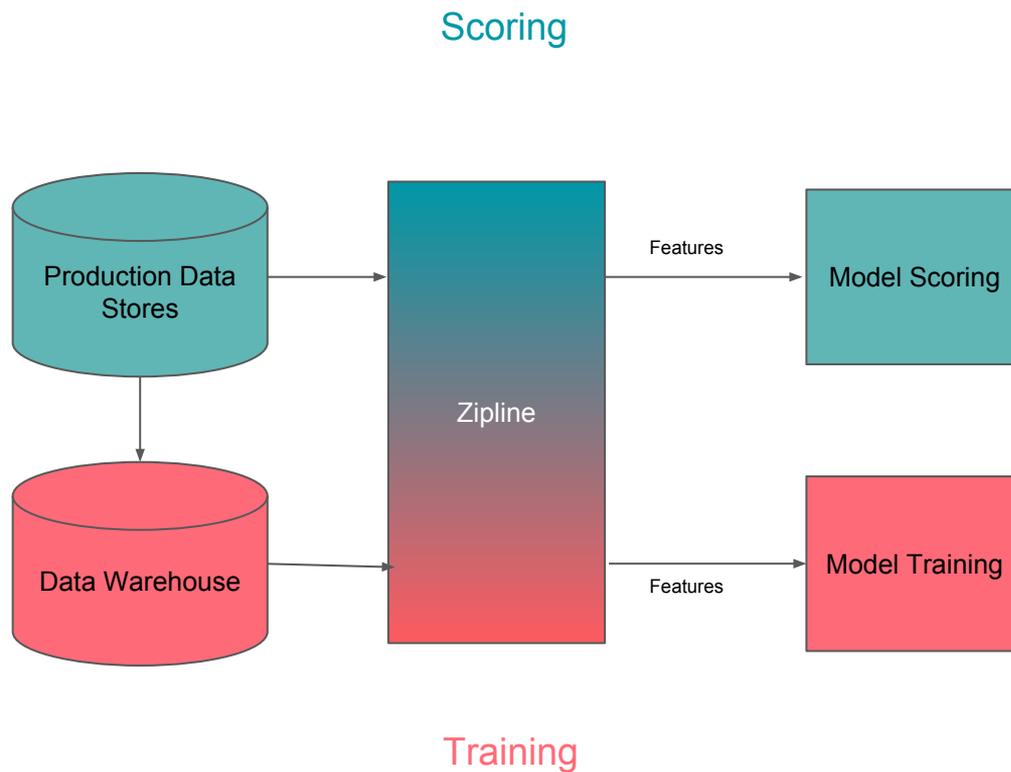
Seamless

- Integration with Deep Thought and ML Automator

Scalable

- Leverages Spark and Flink to scale Batch and Streaming workloads

Zipline Addresses the Consistency Challenge Between Training and Scoring



Big Summary

End-to-End platform to build and deploy ML models to production that is **seamless, versatile, consistent, and scalable**

- Model lifecycle management
- Feature generation & management
- Online & offline inference
- Pipeline library supporting major frameworks
- Docker image customization service
- Multi-tenant training environment

Built on **open source** technology

- TensorFlow, PyTorch, Keras, MXNet, Scikit-learn, XGBoost
- Spark, Jupyter, Kubernetes, Docker, Airflow

To be Open Sourced

We are selecting our first couple private collaborators. If you are interested, please email me at **andrew.hoh@airbnb.com**

Questions?

Appendix

