Feature Store Introduction

Oreintation

"Data is the hardest part of ML and the most important piece to get right. Modelers spend most of their time selecting and transforming features at training time and then building the pipelines to deliver those features to production models. Broken data is the most common cause of problems in production ML systems."

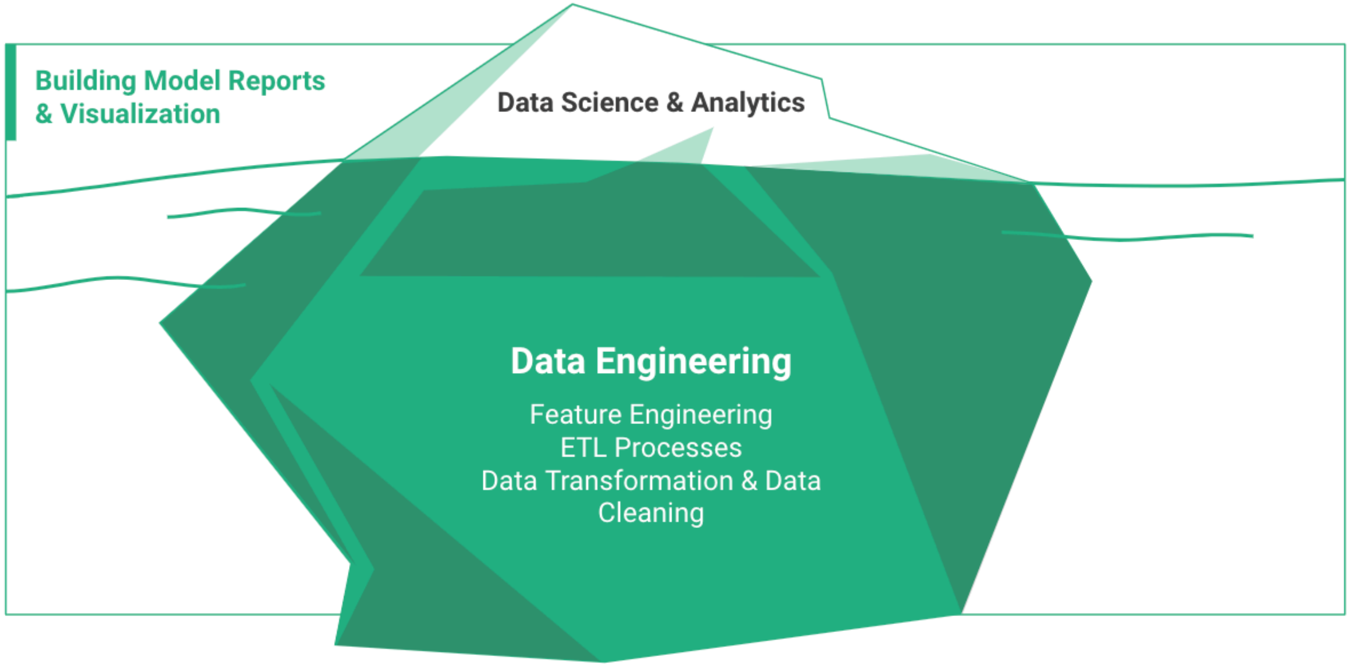
Uber on Michelangelo

What is Feature Store

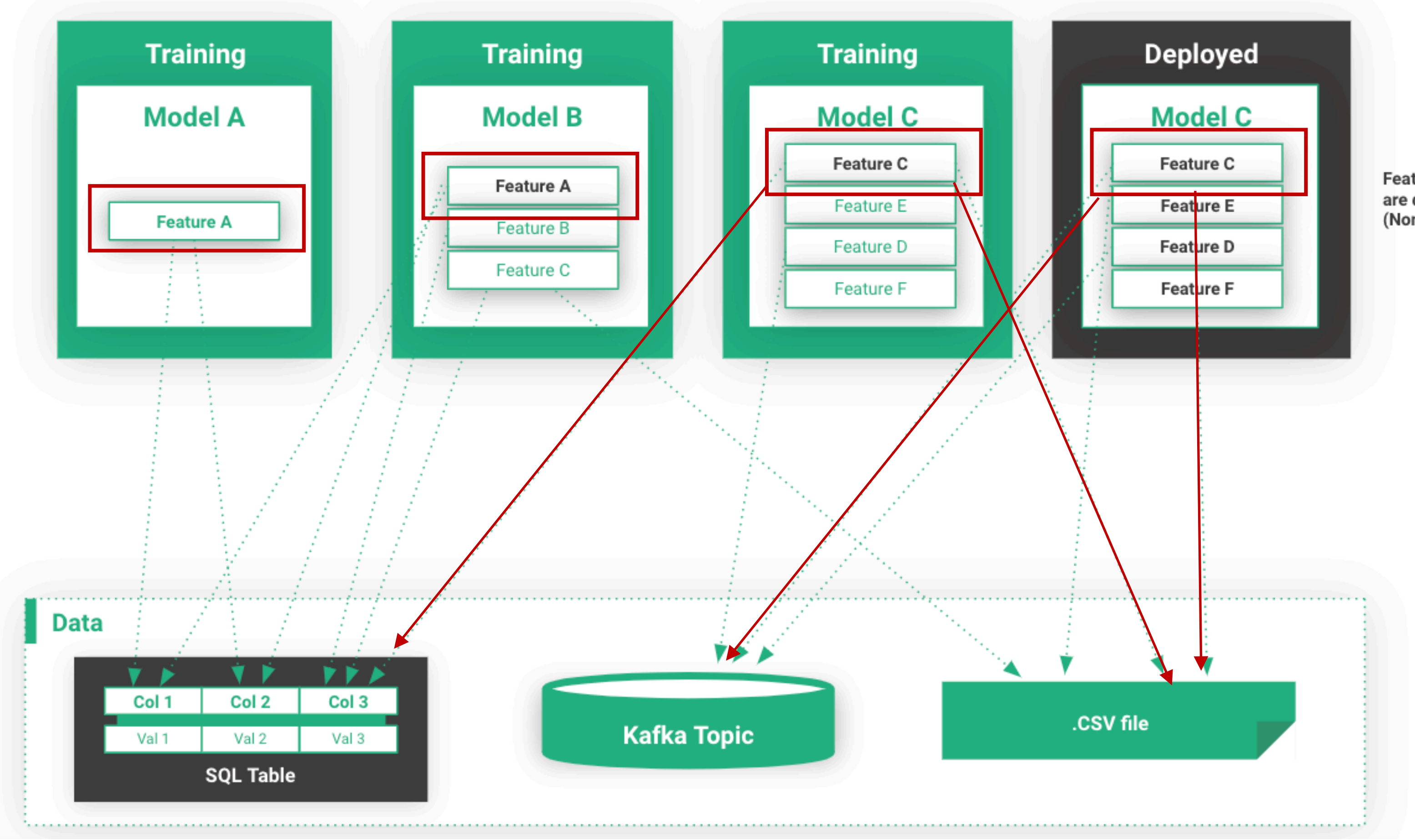
- A data management system for managing machine learning features
- Sharing outputs and assets at all stages in ML pipelines

Why We Need Feature Store

- Ad-hoc feature engineering and training pipelines have a tendency to become complex over time
- **Pipeline Jungle** that is hard to manage



Life Before the Feature Store

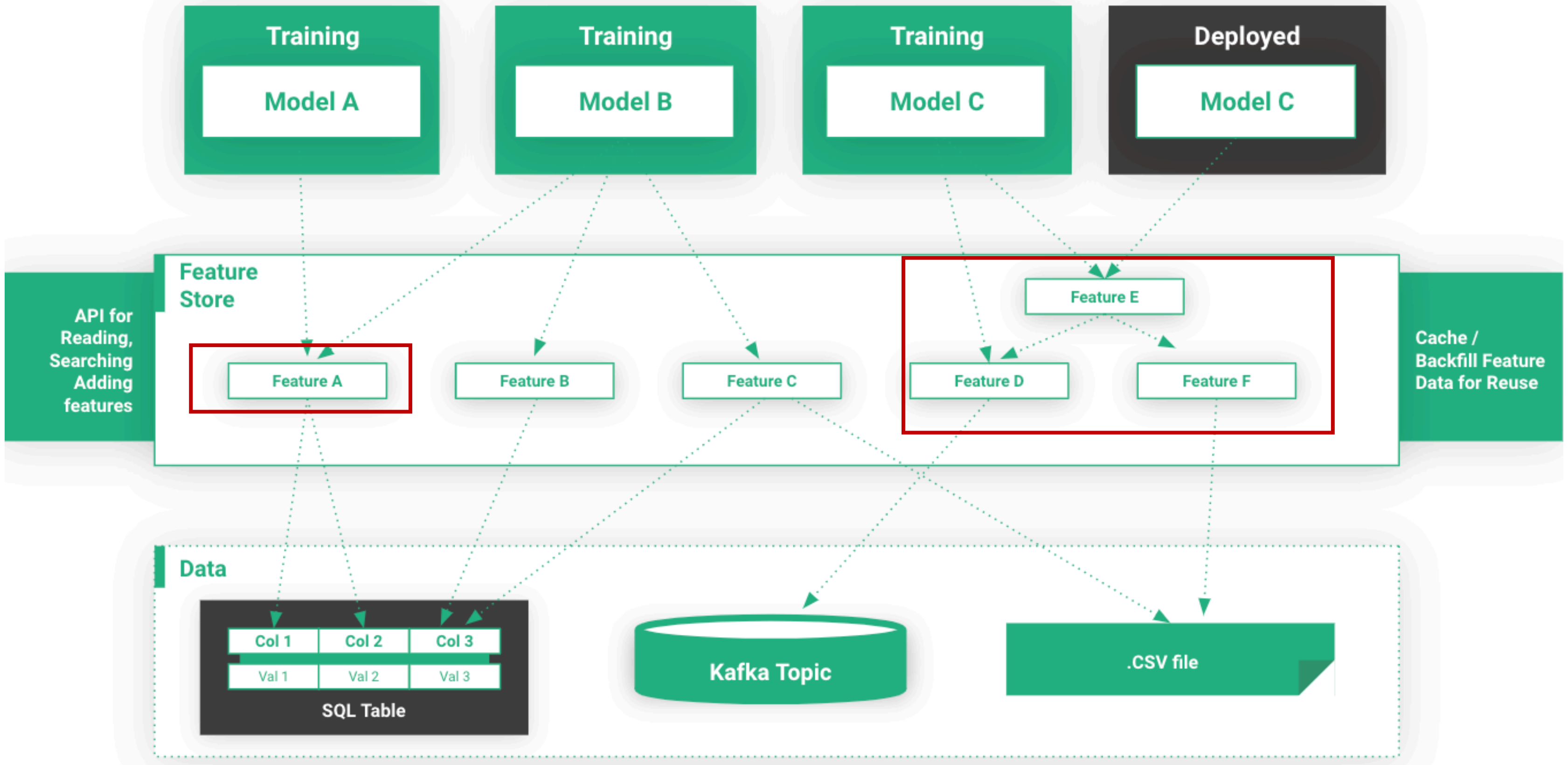


Features in Black
are duplicated code
(Non-DRY)

Without Feature Store

- Lack of feature reuse
- Definitions of features vary
- Inconsistency between training and serving
- ML models can’t access to real-time feature data at low latency and high throughput when they are served in production

With Feature Store



Why Feature Store

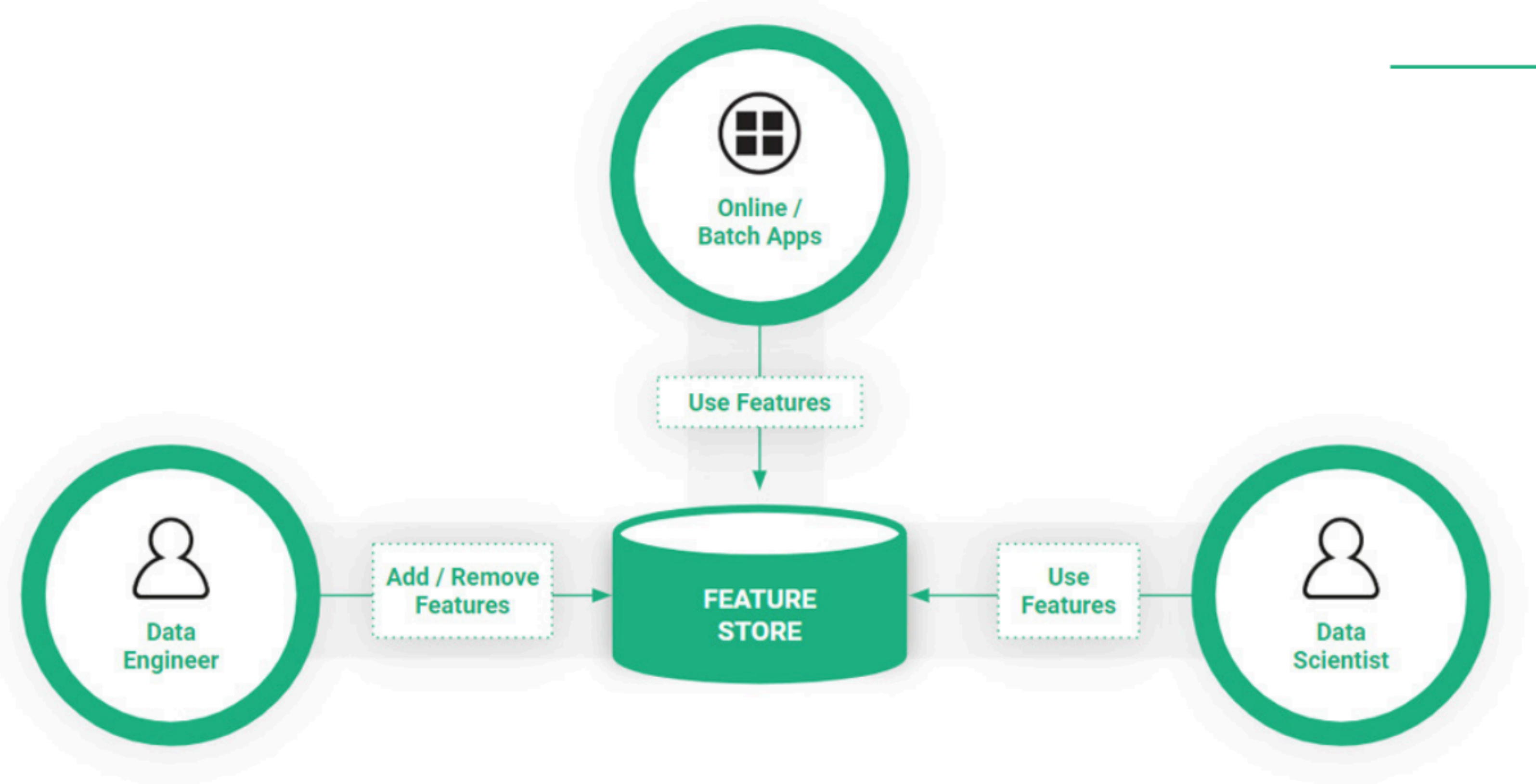
- Provide a unified means of managing feature data from a single person to large enterprises

- Enable discovery, documentation, and insights into your features

- Provide consistent and point-in-time correct access to feature data

- Retrieve historical features for training models

What can Feature Store Do



The API between Data Engineer and Data Scientist

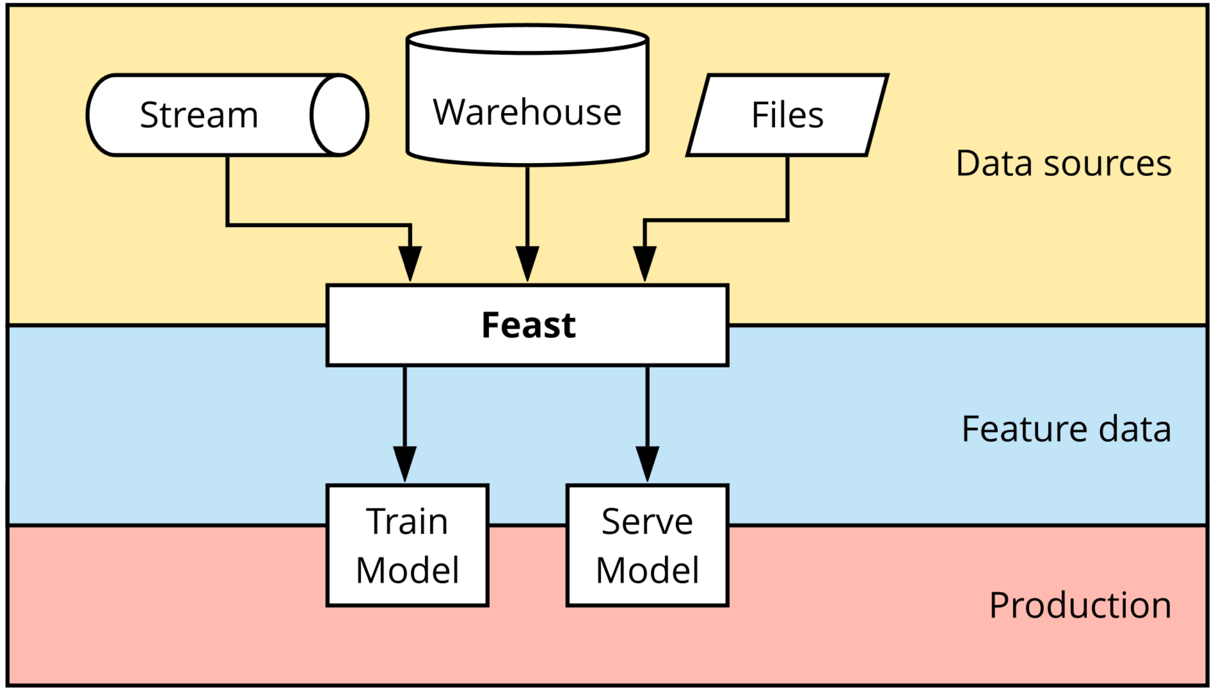
Opensource

FEAST

HOPSWORKS.ai
BY LOGICAL CLOCKS

FEAST

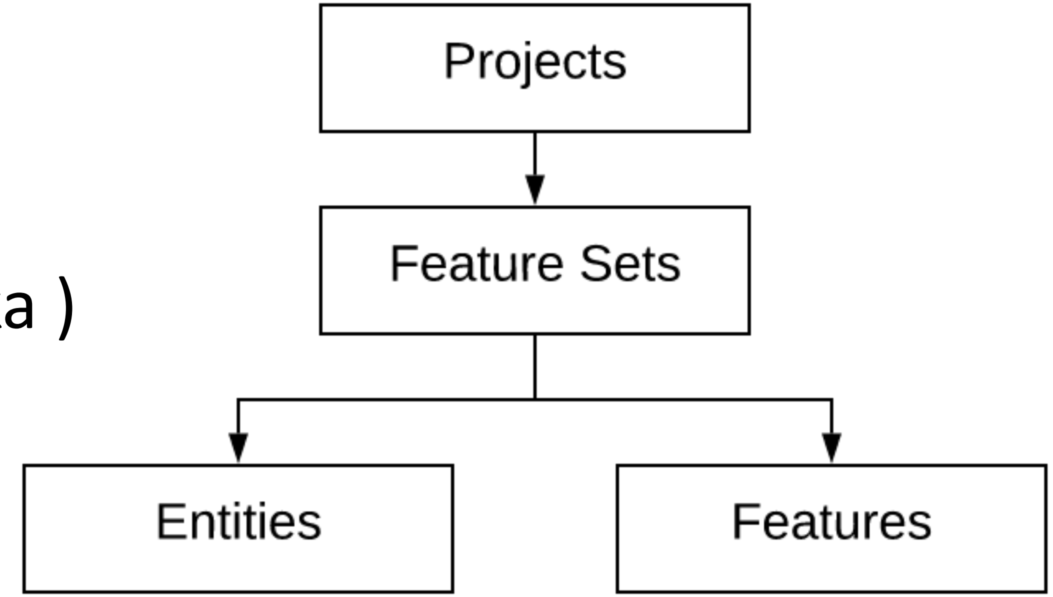
Feast (**Fea**ture **St**ore) is a tool for managing and serving machine learning features.



Concept Hierarchy

FEAST

- Define Feature Set
- Define Entities
- Set Feature
- Link Source for Streaming Data(Kafka)
- Data Ingestion
- Feature retrieval



Feature Set

FEAST

- Specifications that contain both schema and data source information

- Feature sets allow for groups of fields in these data sources to be ingested and store together.

```
## Create Feature set
fs = FeatureSet(
    "overload",
    features = [Feature(name = "Name", dtype=ValueType.STRING,labels=({"name":"English"})),Feature(
    entities = [Entity(name = "ID", dtype = ValueType.INT64)]
)
```

Entity

FEAST

- Entities are used as keys when looking up feature values
- Entities are also used when joining feature values between different feature sets in order to build one large data set to train a model

customer_id.yaml

```yaml
# Entity name
name: customer_id

# Entity value type
value_type: INT64
```

Features

- A feature is an individual measurable property or characteristic of a phenomenon being observed.

```python
from feast import Entity, Feature, ValueType, FeatureSet

# Create a driver entity
driver = Entity("driver_id", ValueType.INT64)

# Create a total trips 24h feature
total_trips_24h = Feature("total_trips_24h", ValueType.INT64)

# Create a feature set with a single entity and a single feature
driver_fs = FeatureSet("driver_fs", entities=[driver], features=[total_trips_24h])

# Register the feature set with Feast
client.apply(driver_fs)
```

Source

- A source(eg: port) that can be used to find feature data
- Currently only supported Kafka

```
print(client.get_feature_set("overload"))
```


```
{
  "spec": {
    "name": "overload",
    "entities": [
      {
        "name": "ID",
        "valueType": "INT64"
      }
    ],
    "features": [
      {
        "name": "Name",
        "valueType": "STRING",
        "labels": {
          "name": "English"
        }
      }
    ],
    "source": {
      "type": "KAFKA",
      "kafkaSourceConfig": {
        "bootstrapServers": "kafka:9092,localhost:9094",
        "topic": "feast-features"
      }
    },
    "project": "default"
  },
  "meta": {
    "createdTimestamp": "2020-08-19T03:11:53Z",
    "status": "STATUS_READY"
  }
}
```

Feast Ingestion

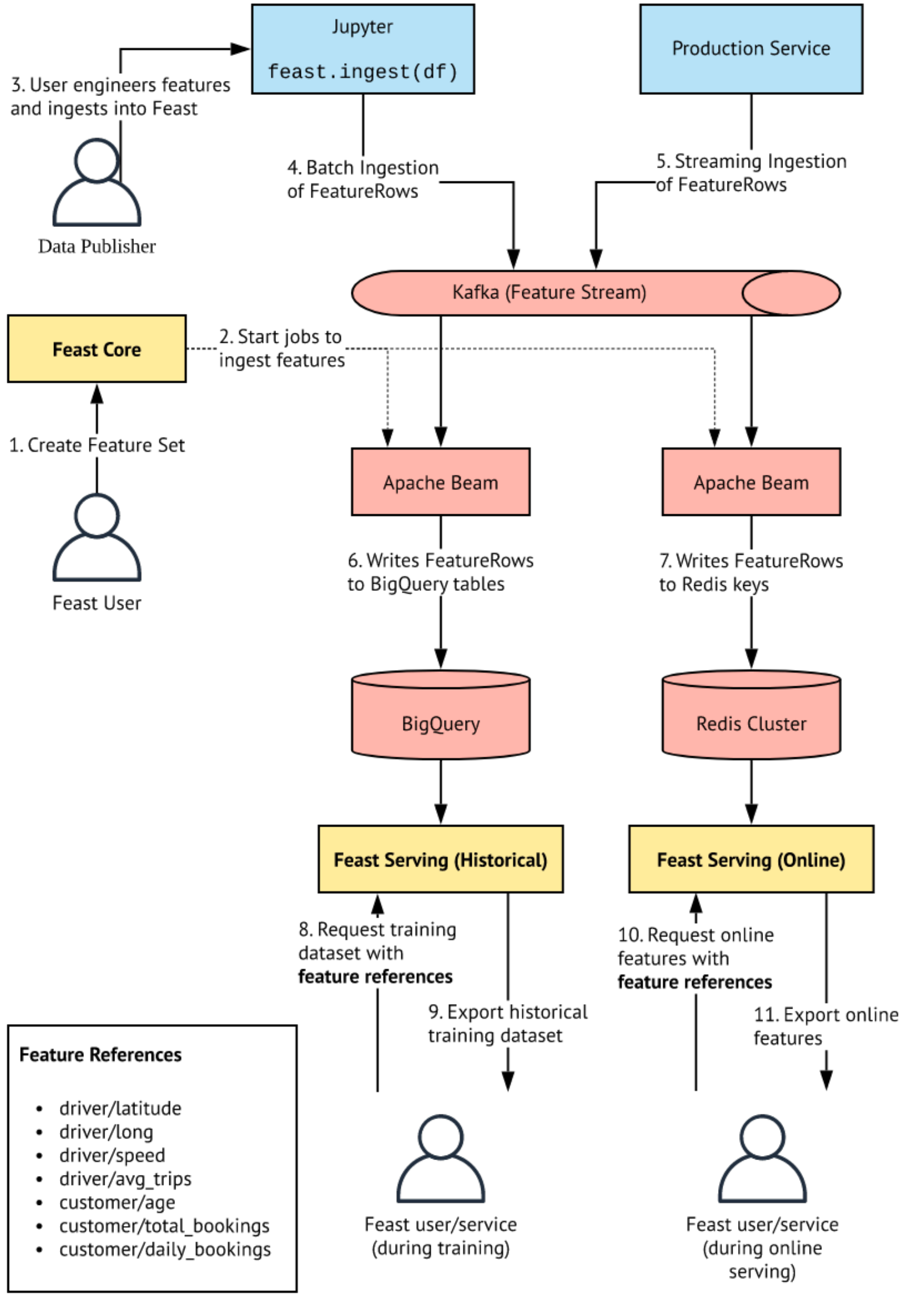
FEAST

- Before user ingests data into Feast, they should register one or more feature sets.
- Once a feature set is registered, Feast will start an Apache Beam job in order to populate a store with data from a source.

```
client.ingest("overload", name)

  0%|          | 0/3 [00:00<?, ?rows/s]

 Waiting for feature set to be ready for ingestion...

 100%|██████████| 3/3 [00:01<00:00,  2.92rows/s]

 Ingestion complete!

 Ingestion statistics:
 Success: 3/3
 Removing temporary file(s)...
```

Architecture

- Historical Feature Serving: BigQuery
- Online Feature Serving : Redis Cluster



Demo

Data Set :Titanic Survival Prediction

- Train.csv : 891 Passenger

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Create New Feature

- Add Datetime Columns
- We split “Fare” into 4,5,6 interval in order to evaluate their model performance
- Create Family Connection
- Create name title Feature
- RF, XGboost, SVM model training
- Evaluate Model
- Check Feature Store Performance

Fare Feature Create

We split “Fare” into “FareBin_Code_4”, “FareBin_Code_5”, “FareBin_Code_6” 3 new feature.

```
## cut Fare data into 4,5,6 class
data['Fare'] = data['Fare'].fillna(data['Fare'].median())

# Making Bins
data['FareBin_4'] = pd.qcut(data['Fare'], 4, labels=[0,1,2,3])
data['FareBin_5'] = pd.qcut(data['Fare'], 5, labels=[0,1,2,3,4])
data['FareBin_6'] = pd.qcut(data['Fare'], 6, labels=[0,1,2,3,4,5])

label = LabelEncoder()
data['FareBin_Code_4'] = label.fit_transform(data['FareBin_4'])
data['FareBin_Code_5'] = label.fit_transform(data['FareBin_5'])
data['FareBin_Code_6'] = label.fit_transform(data['FareBin_6'])

# cross tab
df_4 = pd.crosstab(data['FareBin_Code_4'],data['Pclass'])
df_5 = pd.crosstab(data['FareBin_Code_5'],data['Pclass'])
df_6 = pd.crosstab(data['FareBin_Code_6'],data['Pclass'])

display_side_by_side(df_4,df_5,df_6)
```

Feature Engineering Outcome

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	_5	FareBin_6	FareBin_Code_4	FareBin_Code_5	FareBin_Code_6	Family_size	Connected_Survival	Title1	Title2	datetime
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500		0	0	0	0	2	0.5	Mr	1	2020-08-18 08:56:59.859403
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833		5	3	4	5	2	0.5	Mrs	0	2020-08-18 08:56:59.859403
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250		1	1	1	1	1	0.5	Miss	0	2020-08-18 08:56:59.859403
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000		4	3	4	4	2	0.0	Mrs	0	2020-08-18 08:56:59.859403
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500		1	1	1	1	1	0.5	Mr	1	2020-08-18 08:56:59.859403

Feature Store

FEAST

- Connect to Feast core

```
client = Client(core_url=FEAST_CORE_URL, serving_url=FEAST_ONLINE_SERVING_URL)
```

- Create Titanic feature set

```
## Create Feature set
titanic_fs = FeatureSet(
    "titanic_demo",
    entities = [Entity(name = "PassengerId", dtype = ValueType.INT64)]
)
```

Feature Store

FEAST

- Import Feature to dataset

```
titanic_fs.infer_fields_from_df(train ,replace_existing_features=True)
```

- Import Feature Set with to core

```
client.apply(titanic_fs)
```


```
Feature set updated: "titanic_demo"
{
  "spec": {
    "name": "titanic_demo",
    "entities": [
      {
        "name": "PassengerId",
        "valueType": "INT64"
      }
    ],
    "features": [
      {
        "name": "Ticket",
        "valueType": "STRING"
      },
      {
        "name": "Parch",
        "valueType": "INT64"
      },
```

Feature Store

FEAST

- Ingest data to Feast

```
client.ingest("titanic_demo", train)

  0%|          | 0/891 [00:00<?, ?rows/s]

Waiting for feature set to be ready for ingestion...

100%|██████████| 891/891 [00:01<00:00, 821.55rows/s]

Ingestion complete!

Ingestion statistics:
Success: 891/891
Removing temporary file(s)...
```

Feature Retrieval

- **Feature references:** each feature can be uniquely addressed through a feature reference <Feature Set : Feature>

```
online_features = client.get_online_features(
    feature_refs=[
        "titanic_demo:Sex_index",
        "titanic_demo:Pclass",
        "titanic_demo:FareBin_Code_5",
        "titanic_demo:Connected_Survival",
        "titanic_demo:Title2",
    ],
    entity_rows=temp,
   # entity_rows=[{'PassengerId': 0}],
)
```

Feature Retrieval


```
[fields {
  key: "PassengerId"
  value {
    int64_val: 1
  }
}
fields {
  key: "titanic_demo:Connected_Survival"
  value {
    double_val: 0.5
  }
}
fields {
  key: "titanic_demo:FareBin_Code_5"
  value {
    int64_val: 0
  }
}
fields {
  key: "titanic_demo:Pclass"
  value {
    int64_val: 3
```

Point-in-time-correct Join

FEAST

```
online_features = client.get_online_features(
    feature_refs=[
        "titanic_demo:Sex_index",
        "test1:Pclass",
        "test1:FareBin_Code_5",
        "titanic_demo:Connected_Survival",
        "titanic_demo:Title2",
    ],
    entity_rows=temp,
   # entity_rows=[{'PassengerId': 0}],
)
```

```
[fields {
  key: "PassengerId"
  value {
    int64_val: 1
  }
}
fields {
  key: "test1:FareBin_Code_5"
  value {
    int64_val: 0
  }
}
fields {
  key: "test1:Pclass"
  value {
    int64_val: 3
  }
}
fields {
  key: "titanic_demo:Connected_Survival"
  value {
    double_val: 0.5
```

Future Work

- Feast0.7 future release feast UI
- Retrieve online features for serving models
- Connection with Apache Spark
- Chinese comment for feature definition

Q&A