



The background of the slide features a dark, abstract digital theme. It consists of a grid of small, semi-transparent blue and purple dots representing data points or nodes. Overlaid on this grid are several thin, light-colored lines forming a complex network or circuit board pattern. In the center-left area, there is a cluster of binary code digits ('0's and '1's) in a light blue color. To the right of this cluster, the words 'Build.', 'Unify.', and 'Scale.' are stacked vertically in large, bold, white sans-serif font. The 'U' in 'Unify.' is colored green, while the other letters are white.

Build.  
Unify.  
Scale.

WIFI SSID:SparkAISummit | Password: UnifiedAnalytics

ORGANIZED BY  
 databricks



# Utilizing MLFlow and Kubernetes to build an Enterprise ML Platform

Nick Pinckernell, Comcast Applied AI Research

#UnifiedAnalytics #SparkAIsummit

# Topics

| TOPIC                                                              | WHY?                                                                            |
|--------------------------------------------------------------------|---------------------------------------------------------------------------------|
| Example of data pipeline abstraction                               | Modular components and reuse are important for abstracting complex systems      |
| Ways to package and track ML project and experiments               | Consistency and reproducibility is key for scale                                |
| How Comcast uses Kubeflow to serve and deploy models and pipelines | A tangible example to help you brainstorm about your organizations requirements |

# Challenges and motivations

- Before, there was no
  - model management or tracking
  - standardization for model packaging or deployments
- Cumbersome deployment process
  - Deployment required code rewrite from research to operations
  - Days or weeks to deploy
- Response and tradeoff: restrict model complexity

# Requirements

Minimum requirements from our organization

- Zero code refactoring or rewriting between research ready models and production
- Easier experiment and model tracking
- Researchers need to deploy their own models
- A/B testing for quick model enhancement testing in production
- Ability to modularize and inject custom metrics and workflows at each step

# Solution – existing technologies



# Data pipeline abstraction

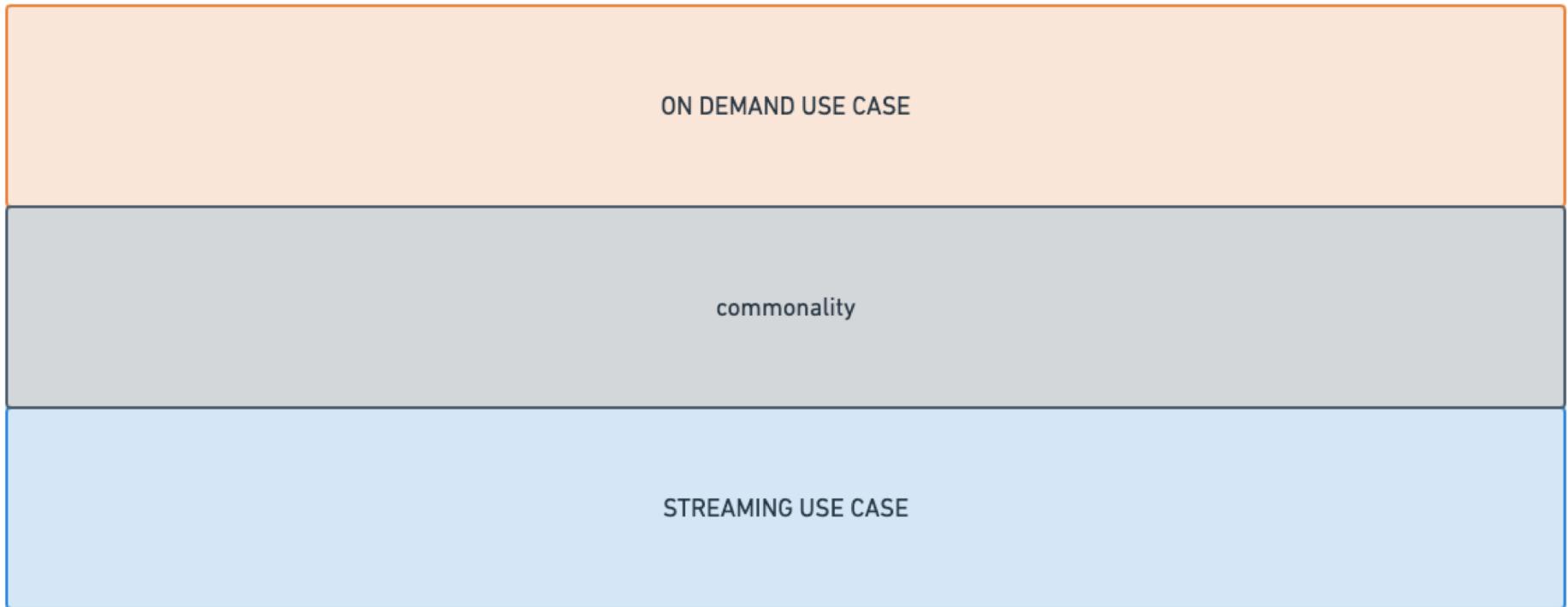
- Determine use cases
- Identify commonalities for modularization
- Abstract interfaces
- Automate configuration

# Pipeline abstraction

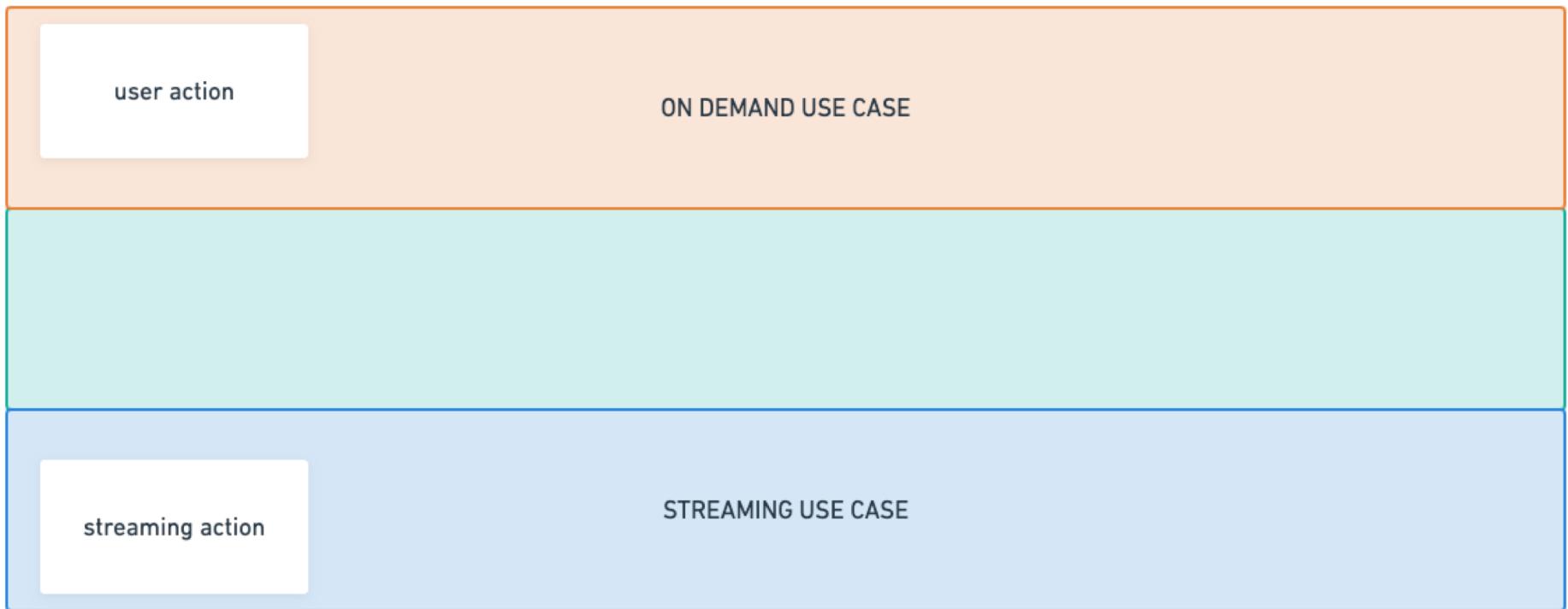
ON DEMAND USE CASE

STREAMING USE CASE

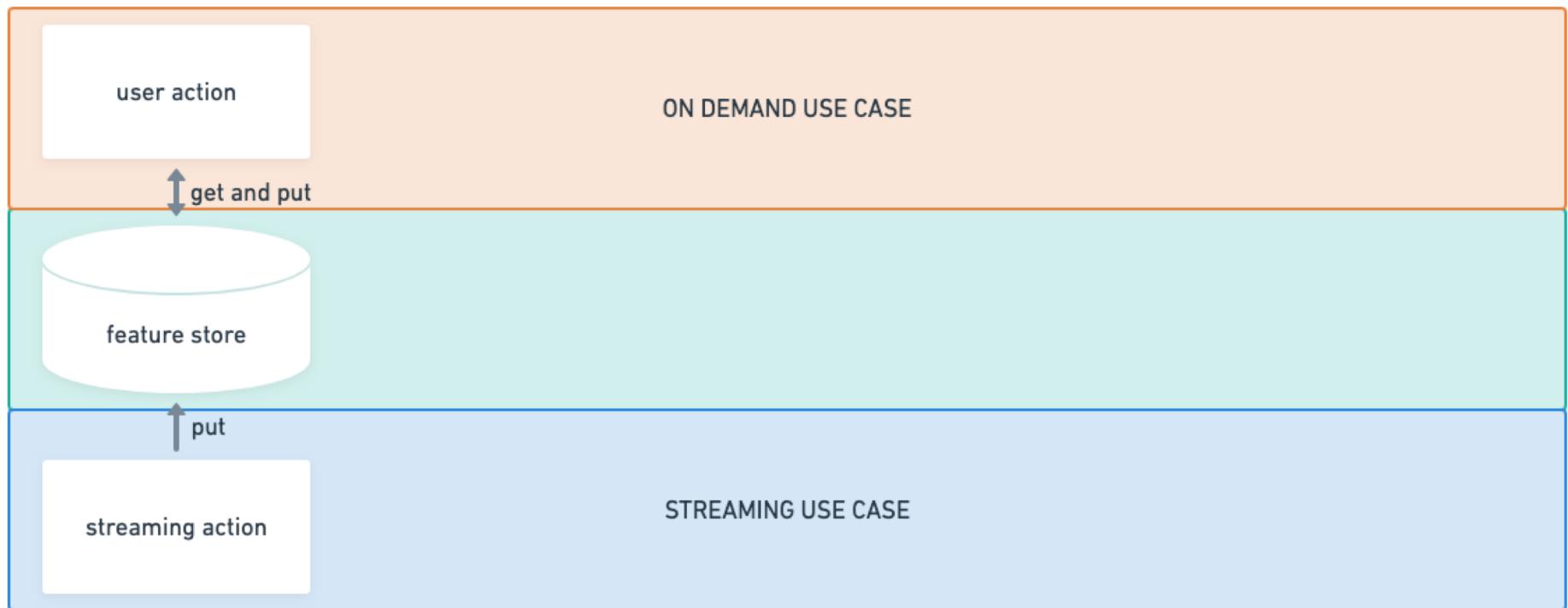
# Pipeline abstraction



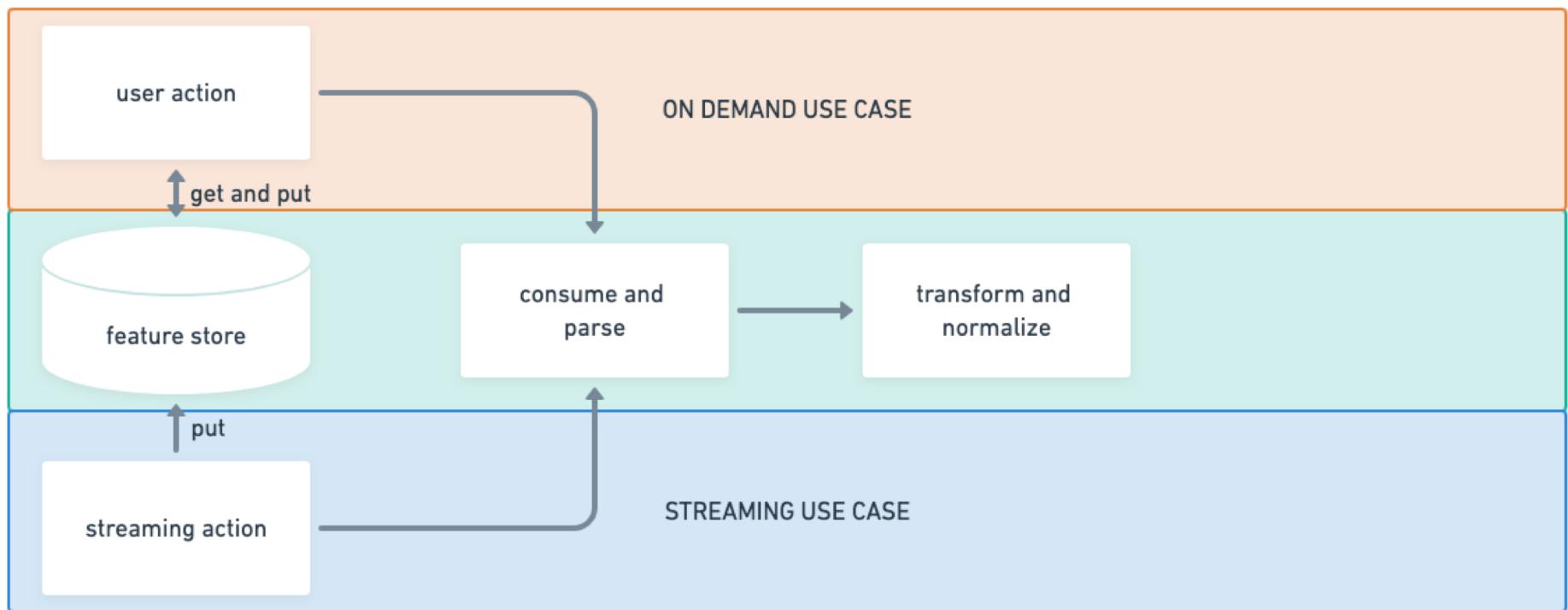
# Pipeline abstraction



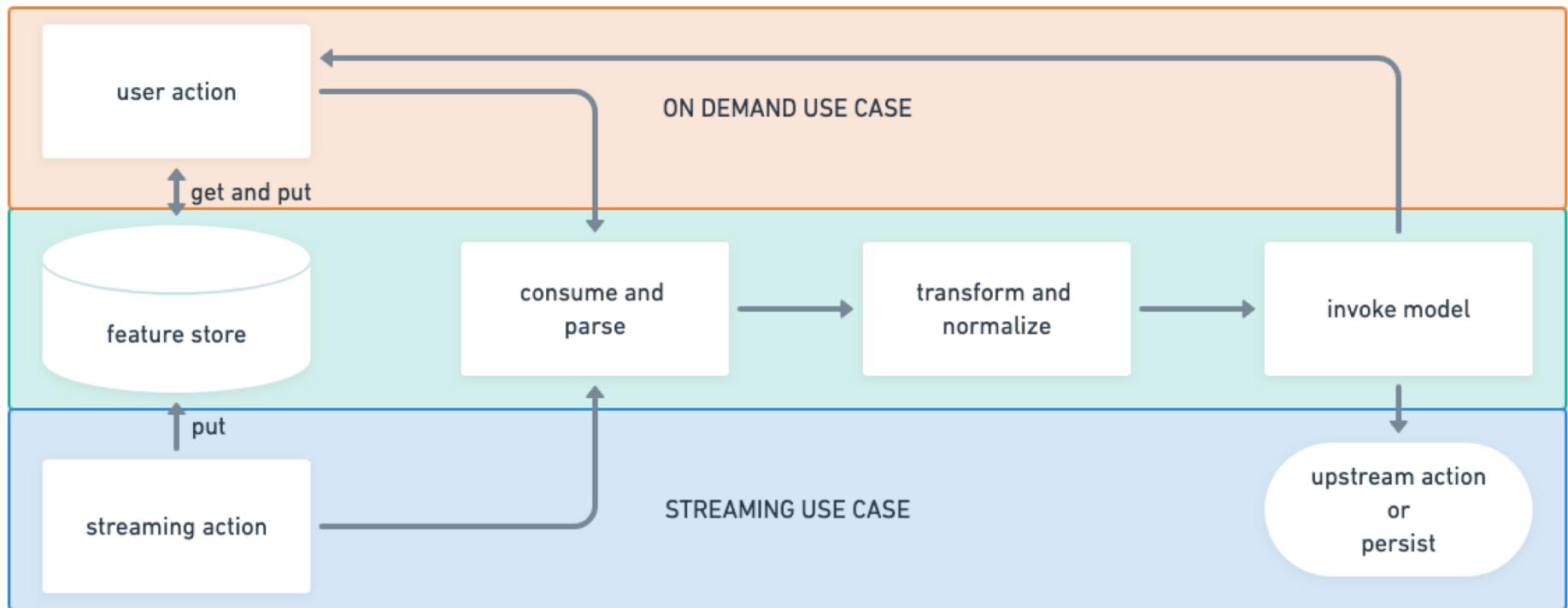
# Pipeline abstraction



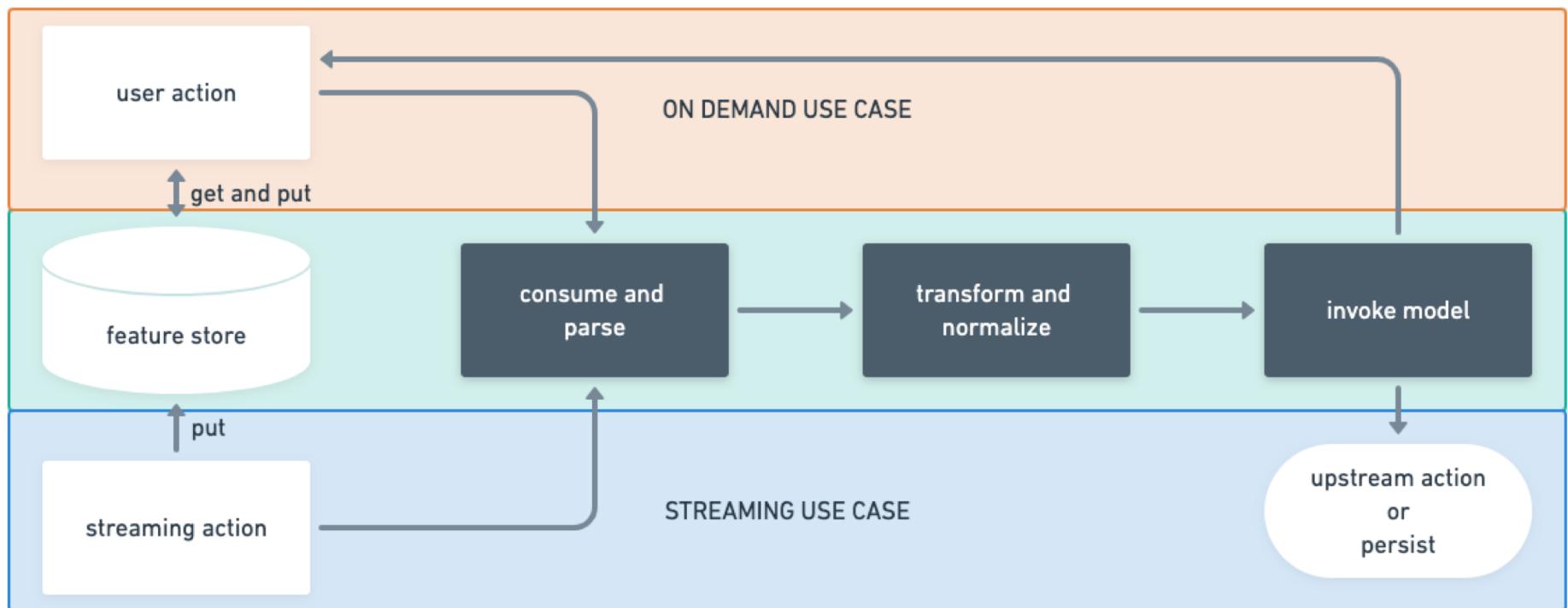
# Pipeline abstraction



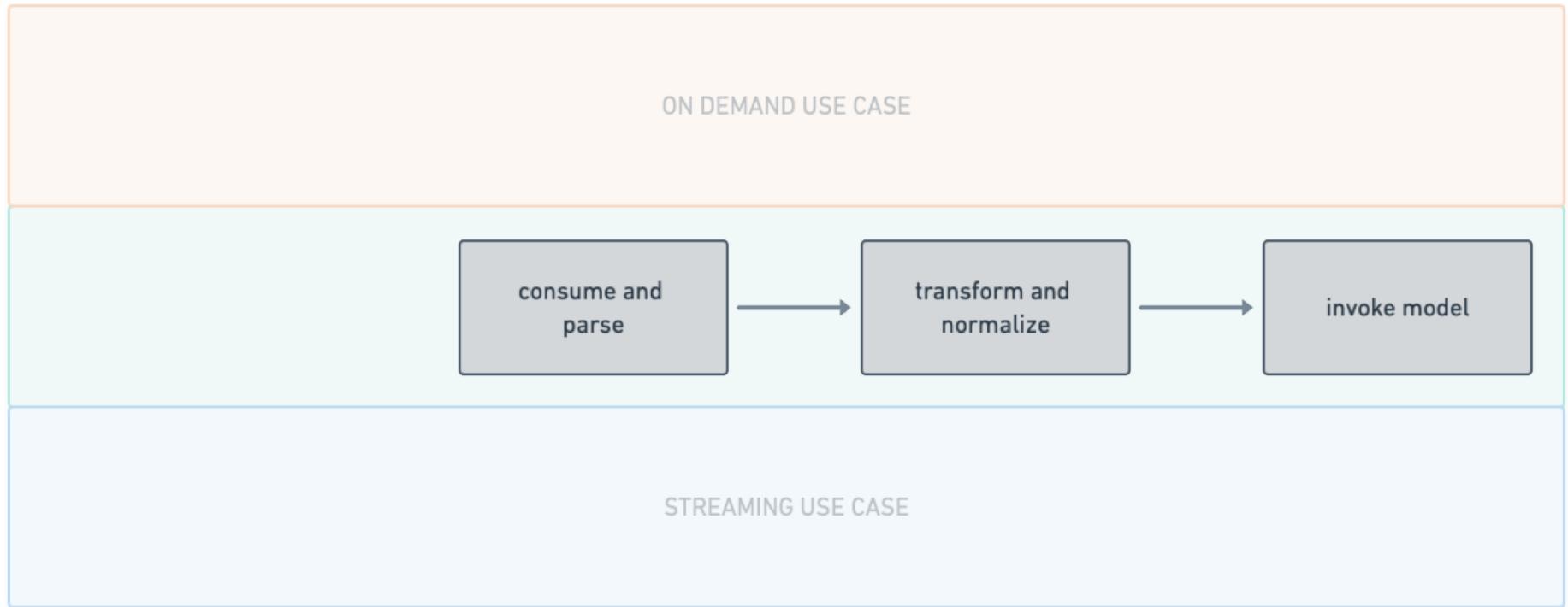
# Pipeline abstraction



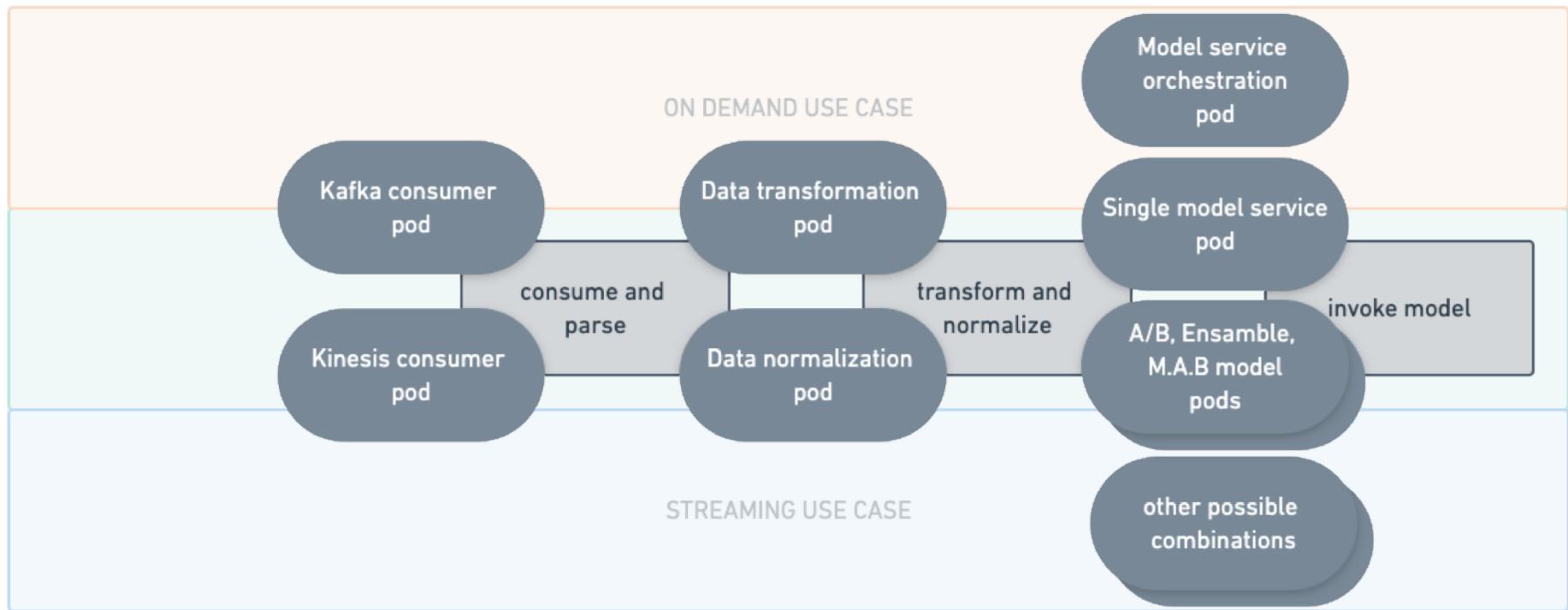
# Pipeline abstraction



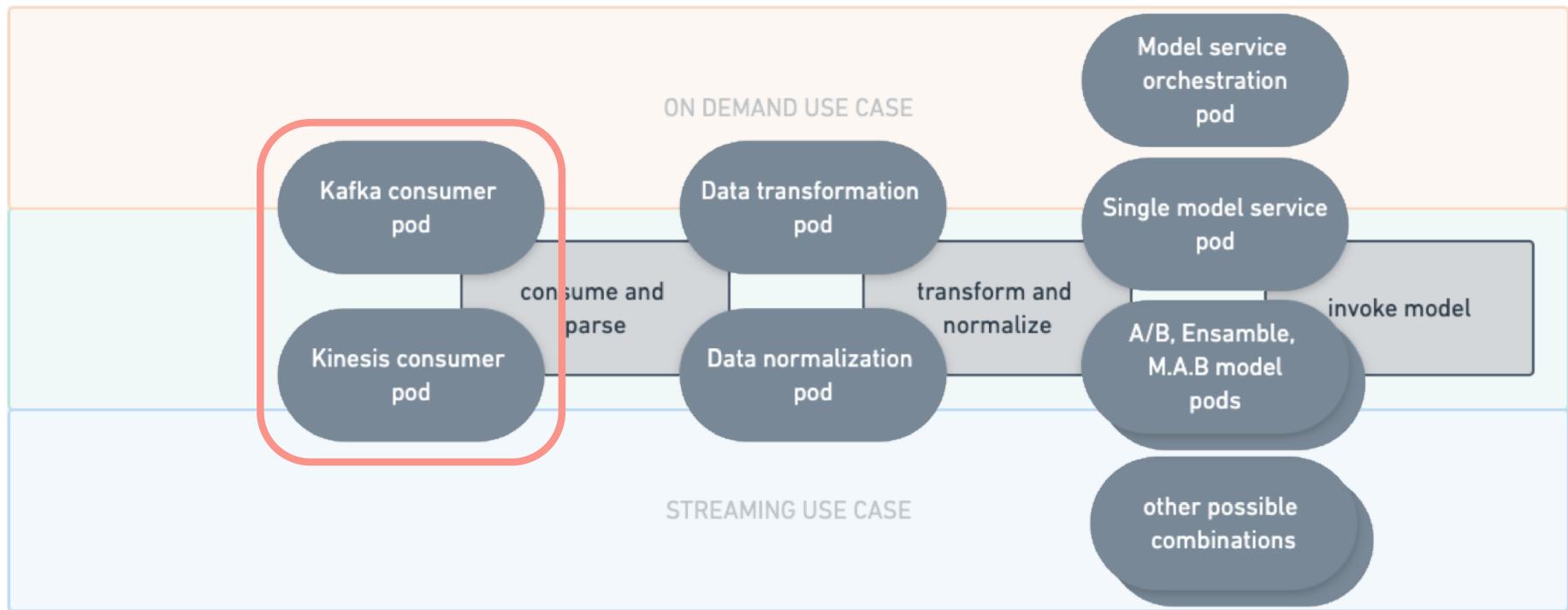
# Pipeline abstraction



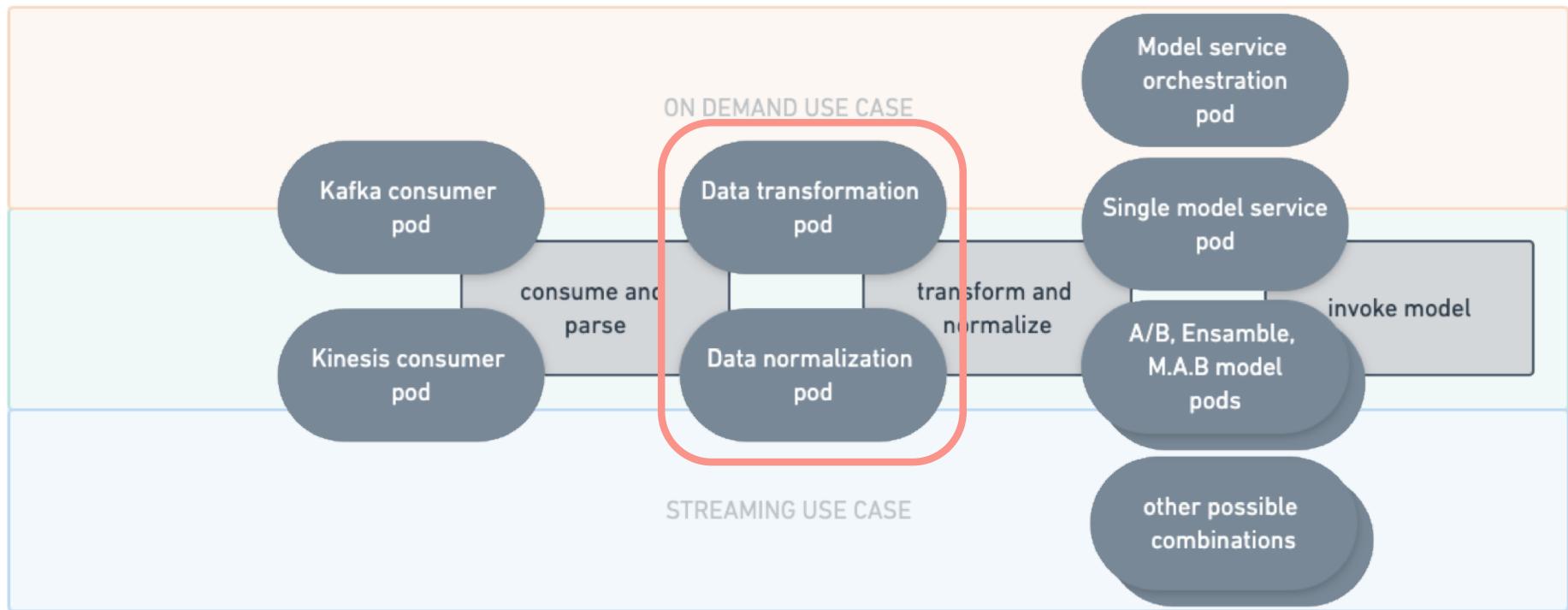
# Pipeline abstraction



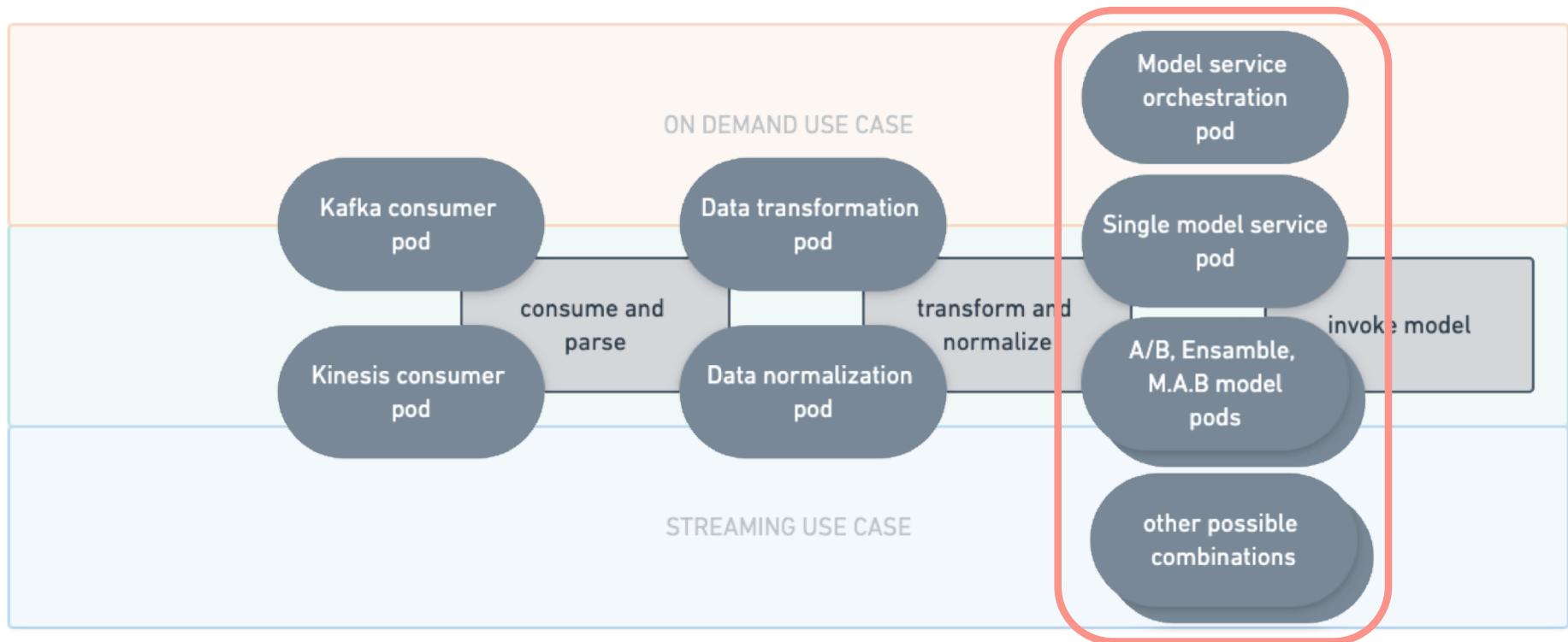
# Pipeline abstraction



# Pipeline abstraction



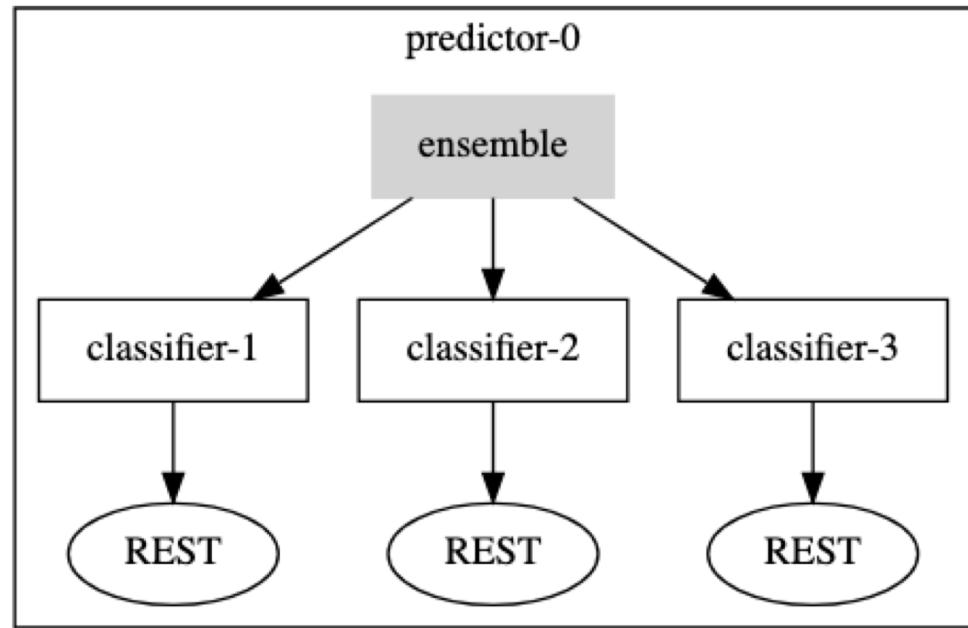
# Pipeline abstraction



# Seldon inference graphs

Allows for complex graphs

- A/B testing
- Ensembles
- Multi-armed bandit
- Custom combinations



[https://github.com/SeldonIO/seldon-core/blob/release-0.2/notebooks/advanced\\_graphs.ipynb](https://github.com/SeldonIO/seldon-core/blob/release-0.2/notebooks/advanced_graphs.ipynb)

# Packaging and tracking

1. Researchers code and train models with Databricks, Spark
  2. Experiments tracked with MLFlow
  3. Packaging and model tracking with MLFlow and Kubeflow
- 
- MLFlow standard packaging formats
    - scikit-learn
    - h2o
    - TensorFlow
    - more

# An MLFlow experiment

```
23  # Start a MLFlow experiment and label it
24  with mlflow.start_run(run_name="no outliers, default hyperparams"):
25      # train
26      clf = train(train_x, train_y, solver, C, multi_class)
27      # predict
28      predict = clf.predict(test_x)
29      # eval metrics
30      rmse, mae, r2 = eval_metrics(test_y, predict)
31
32      # log some params
33      mlflow.log_param("c", C)
34      mlflow.log_param("multi_class", multi_class)
35      # log some metrics
36      mlflow.log_metric("rmse", rmse)
37      mlflow.log_metric("r2", r2)
38      mlflow.log_metric("mae", mae)
39
40      # finally log the model and end
41      mlflow.sklearn.log_model(clf, "model")
42      mlflow.end_run()
```

# MLFlow – multiple experiments

The screenshot shows the MLflow interface on a Databricks workspace. The left sidebar includes icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search, with 'MLflow' selected. The main area displays the experiment details for '/Users/Nick@comcast/iris\_outliers'. The experiment ID is 1818539 and the artifact location is dbfs:/databricks/mlflow/1818539. A search bar filters runs by 'metrics.rmse < 1 and params.model = "tree"'. The results table shows four matching runs, each with a checkbox, date, user, run name, source, version, and parameter and metric values. The table has columns for Parameters and Metrics.

|                          | Date                | User | Run Name                            | Source                        | Version | a_dataset | a_outliers | c           | multi_class | solver | mae   | r2    | rmse |
|--------------------------|---------------------|------|-------------------------------------|-------------------------------|---------|-----------|------------|-------------|-------------|--------|-------|-------|------|
| <input type="checkbox"/> | 2019-03-29 23:46:05 | Nick | with outliers, default hyperparams  | <a href="#">iris_outliers</a> | 2       | 1         | 1.0        | ovr         | liblinear   | 0.086  | 0.851 | 0.317 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:07 | Nick | no outliers, default hyperparams    | <a href="#">iris_outliers</a> | 1       | 0         | 1.0        | ovr         | liblinear   | 0.289  | 0.571 | 0.537 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:10 | Nick | with outliers, modified hyperparams | <a href="#">iris_outliers</a> | 2       | 1         | 100000.0   | multinomial | lbfgs       | 0.065  | 0.883 | 0.281 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:13 | Nick | no outliers, modified hyperparams   | <a href="#">iris_outliers</a> | 1       | 0         | 100000.0   | multinomial | lbfgs       | 0.067  | 0.901 | 0.258 |      |

# MLFlow – multiple experiments

The screenshot shows the MLflow interface on a Databricks platform. The left sidebar includes icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search, with 'MLflow' selected. The main area displays the experiment details for '/Users/Nick@comcast/iris\_outliers'. The Experiment ID is 1818539 and the Artifact Location is dbfs:/databricks/mlflow/1818539. A search bar filters runs by 'metrics.rmse < 1 and params.model = "tree"'. The results table shows four matching runs, each with a checkbox, Date, User, Run Name, Source, Version, Parameters, and Metrics. The metrics columns include mae, r2, and rmse. The last row's 'rmse' value, 0.258, is circled in red.

|                          | Date                | User | Run Name                            | Source                        | Version | a_dataset | a_outliers | c           | multi_class | solver | mae   | r2    | rmse |
|--------------------------|---------------------|------|-------------------------------------|-------------------------------|---------|-----------|------------|-------------|-------------|--------|-------|-------|------|
| <input type="checkbox"/> | 2019-03-29 23:46:05 | Nick | with outliers, default hyperparams  | <a href="#">iris_outliers</a> | 2       | 1         | 1.0        | ovr         | liblinear   | 0.086  | 0.851 | 0.317 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:07 | Nick | no outliers, default hyperparams    | <a href="#">iris_outliers</a> | 1       | 0         | 1.0        | ovr         | liblinear   | 0.289  | 0.571 | 0.537 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:10 | Nick | with outliers, modified hyperparams | <a href="#">iris_outliers</a> | 2       | 1         | 100000.0   | multinomial | lbfgs       | 0.065  | 0.883 | 0.281 |      |
| <input type="checkbox"/> | 2019-03-29 23:46:13 | Nick | no outliers, modified hyperparams   | <a href="#">iris_outliers</a> | 1       | 0         | 100000.0   | multinomial | lbfgs       | 0.067  | 0.901 | 0.258 |      |

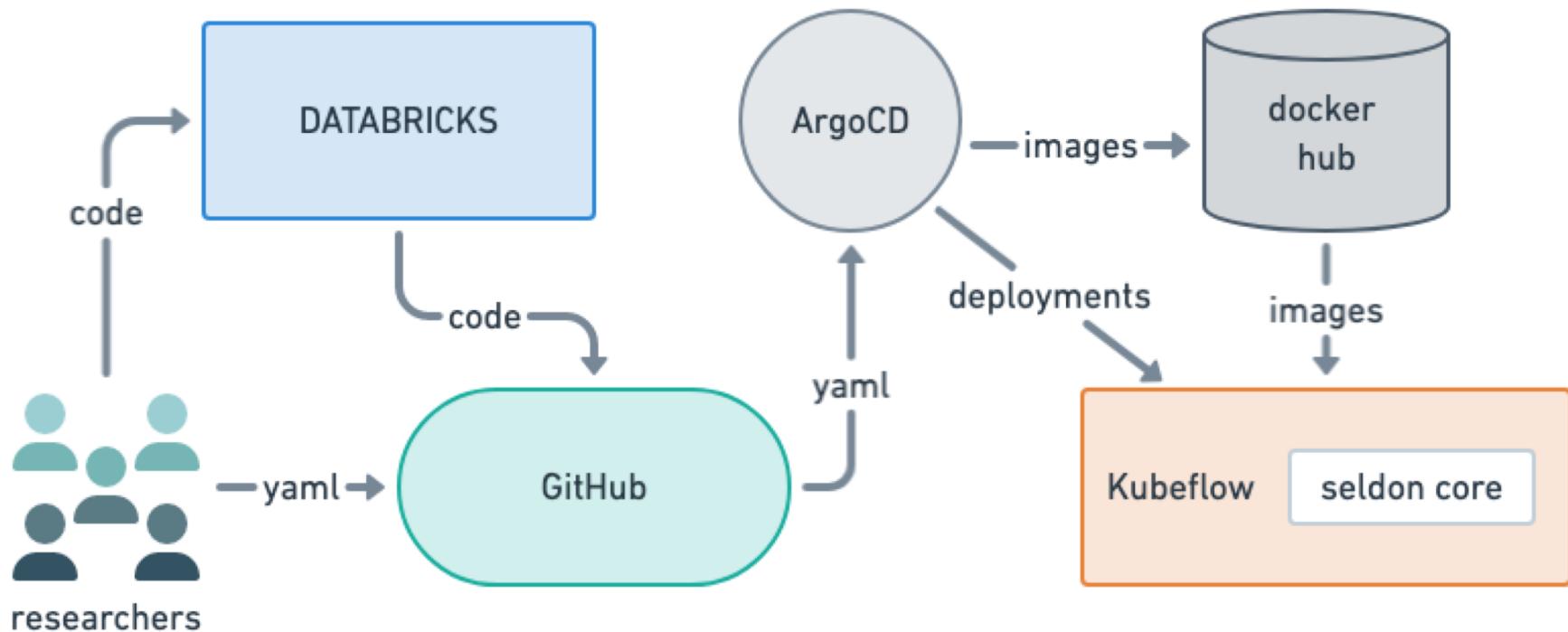
# MLFlow packaging

The screenshot shows the Databricks MLflow interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, and Jobs. The 'MLflow' tab is selected. The main area displays a run details page for a specific run:

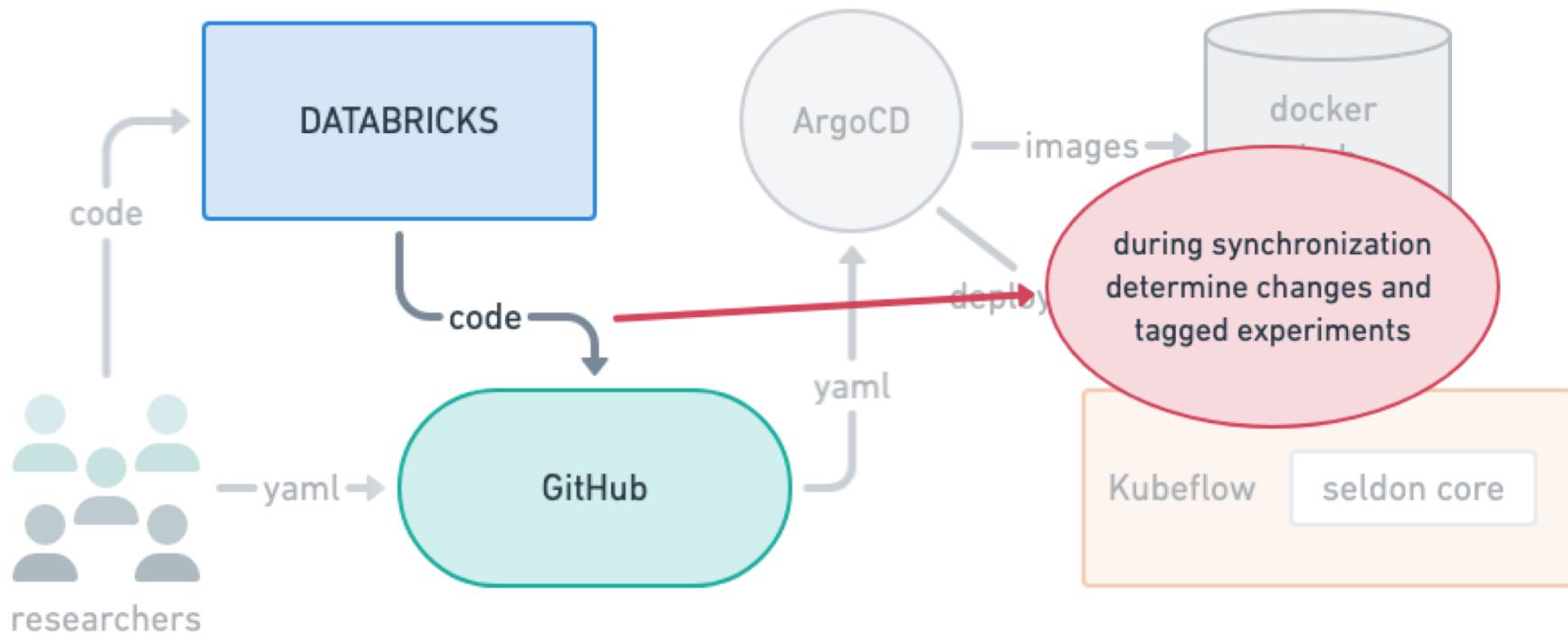
- Path:** /Users/Nick@comcast/iris\_outliers > no outliers, modified hyperparams
- Date:** 2019-03-29 23:46:13
- Source:** iris\_outliers
- Duration:** 2.5s
- Run ID:** 20ae6c89e5a843beb17dd9e70de62b58 (circled in red)
- User:** Nick@comcast (circled in red)
- Notes:** None
- Parameters:**

| Name       | Value |
|------------|-------|
| a_dataset  | 1     |
| a_outliers | 0     |

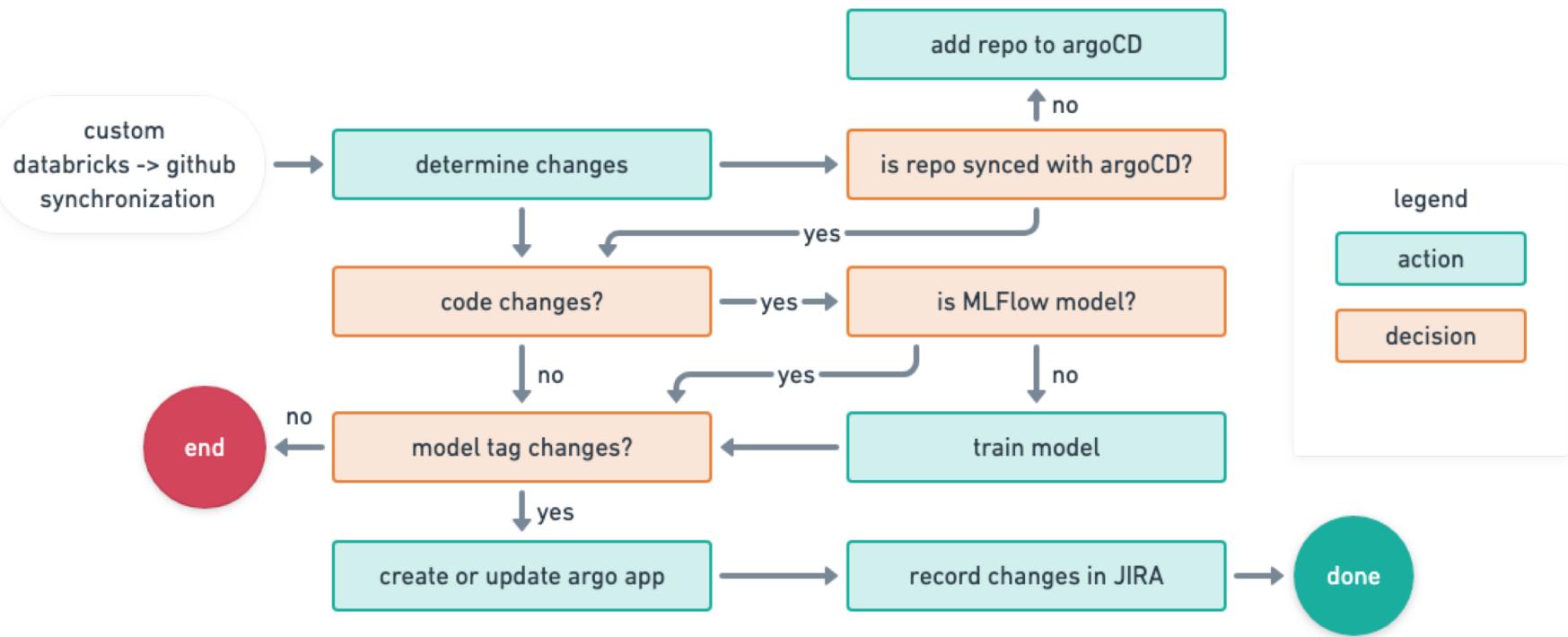
# Research and model flow



# Research and model flow



# Research and model flow – at scale



# Model serving with Kubeflow

Considerations and requirements

- Resilient
- Highly available
- Rate limiting
- Shadow deployments
- Auto-scaling (WIP)



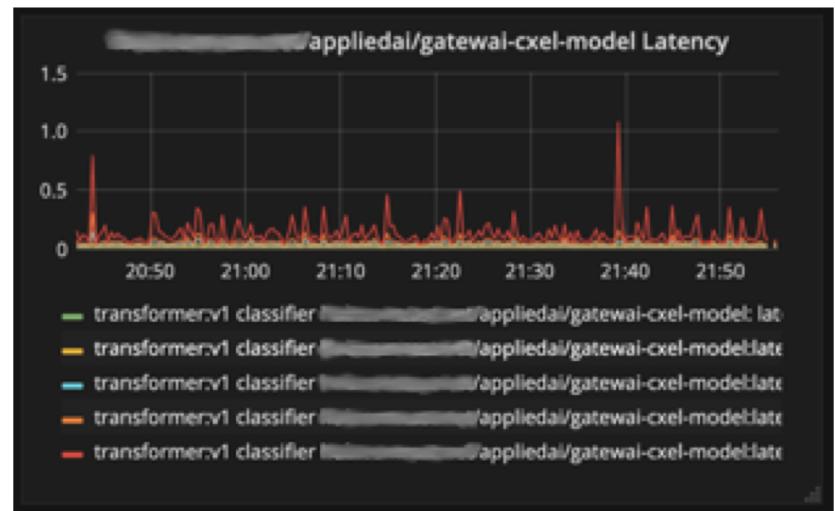
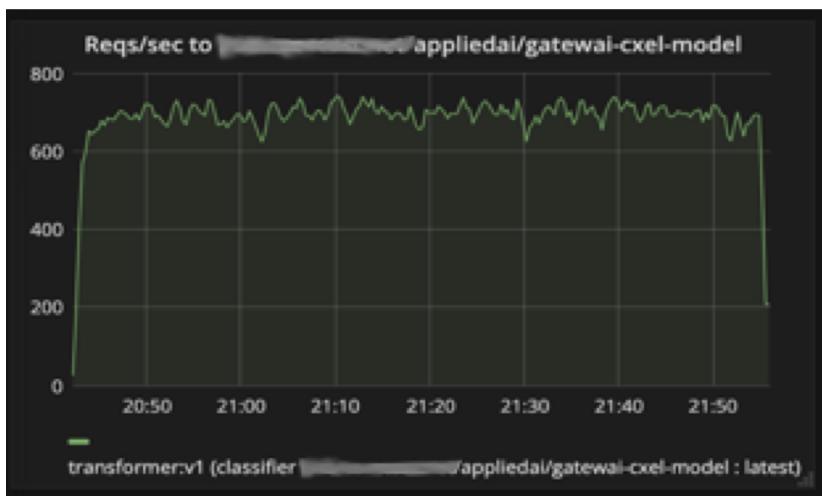
Ambassador

<http://www.getambassador.io>

# Throughput

Static number of replicas Determined after

- Constant and burst load testing with Locust



# DEMO

A demonstration of

- MLFlow experiments
  - Serving the chosen model
- Implementation of components
  - Consumer pod
  - Model pod
  - Producer logic (to simulate real requests)

# Choosing the run

## sklearn\_iris

Experiment ID: 1      Artifact Location: /Users/nick/sais\_demo/sklearn\_iris\_mlflow/mlruns/1

Search Runs: metrics.rmse < 1 and params.model = "tree"      State: Active ▾      Search

Filter Params: alpha, lr      Filter Metrics: rmse, r2      Clear

4 matching runs

Compare

Delete

Download CSV 



|                                                                                     | Date                   | User | Run Name                            | Source                                                                                            | Version | Parameters |            |             |             |        | Metrics |       |       |      |
|-------------------------------------------------------------------------------------|------------------------|------|-------------------------------------|---------------------------------------------------------------------------------------------------|---------|------------|------------|-------------|-------------|--------|---------|-------|-------|------|
|                                                                                     |                        |      |                                     |                                                                                                   |         | a_dataset  | a_outliers | c           | multi_class | solver | acc     | mae   | r2    | rmse |
|    | 2019-04-19<br>22:03:23 | nick | no outliers, default hyperparams    |  train_iris.py   | 1       | 0          | 1.0        | ovr         | liblinear   | 0.711  | 0.289   | 0.571 | 0.537 |      |
|   | 2019-04-19<br>22:03:24 | nick | with outliers, default hyperparams  |  train_iris.py  | 2       | 1          | 1.0        | ovr         | liblinear   | 0.783  | 0.231   | 0.618 | 0.509 |      |
|  | 2019-04-19<br>22:03:24 | nick | no outliers, modified hyperparams   |  train_iris.py | 1       | 0          | 100000.0   | multinomial | lbfgs       | 0.941  | 0.059   | 0.912 | 0.243 |      |
|  | 2019-04-19<br>22:03:24 | nick | with outliers, modified hyperparams |  train_iris.py | 2       | 1          | 100000.0   | multinomial | lbfgs       | 0.916  | 0.098   | 0.814 | 0.355 |      |

# Choosing the model

**mlflow**

**sklearn\_iris > no outliers, modified hyperparams** ▾

Date: 2019-04-19 22:03:24  
Source: train\_iris.py  
Duration: 170ms

Run ID: **70fec8012099461b80766a6ca098cd31**    
User: nick

▼ Notes 

*None*

▼ Parameters

| Name        | Value       |
|-------------|-------------|
| a_dataset   | 1           |
| a_outliers  | 0           |
| c           | 100000.0    |
| multi_class | multinomial |
| solver      | lbfgs       |

```

from mlflow import pyfunc
# import os
import pandas as pd

class IrisClassifier(object):

    def __init__(self):
        # instead of loading the saved model, use a specific MLFlow run
        # self.model = joblib.load('IrisClassifier.sav')
        # self.pyfunc_model = pyfunc.load_pyfunc("mlruns/0/" +
        # next(os.walk('mlruns/0'))[1][0] + "/artifacts/model")
        self.pyfunc_model = pyfunc.load_pyfunc(
            "mlruns/1/70fec8012099461b80766a6ca098cd31/artifacts/model")

    def predict(self, X, features_names):
        # instead of calling the saved model
        # return self.model.predict_proba(X)

        # call the MLFlow model from the run we liked the best
        if not features_names is None and len(features_names)>0:
            df = pd.DataFrame(data=X, columns=features_names)
        else:
            df = pd.DataFrame(data=X)

        return self.pyfunc_model.predict(df)

```

# Implementing the model



# Implementing the consumer

```
from kafka import KafkaConsumer
import numpy as np
import requests

bootstrap_server = 'host.docker.internal:9092'
# create kafka consumer for latest messages
consumer = KafkaConsumer('mnist-topic',
                         group_id='test-group',
                         bootstrap_servers=[bootstrap_server])

# ambassador
url = 'http://seldon-core-ambassador/seldon/seldon-sklearn-iris-mlflow-dep/api/v0.1/predictions'
for message in consumer:
    features_and_names = np.frombuffer(message.value, dtype=np.dtype(('U', 32)))
    names = features_and_names[0:4]
    features = features_and_names[4:8]

    jsonData = {"data": {"names": names.tolist(),
                         "tensor": {"shape": [1, 4], "values": features.tolist()}}}
    r = requests.post(url, json=jsonData)
```

# Implementing the producer

```
from kafka import KafkaProducer
import numpy as np
from sklearn.datasets import load_iris
from time import sleep

# start our producer
producer = KafkaProducer(bootstrap_servers=['localhost:9092'])

# load the iris data set
d = load_iris()

# loop forever and publish 2 msgs per sec
while True:
    random_row = d.data[np.random.randint(len(d.data))]
    short_feature_names = list(map(lambda name:
        str.split(name)[0]+'_'+str.split(name)[1], d.feature_names))
    features_and_names = np.array([short_feature_names, random_row])

    future = producer.send('mnist-topic', features_and_names.tobytes())
    sleep(0.5)
```

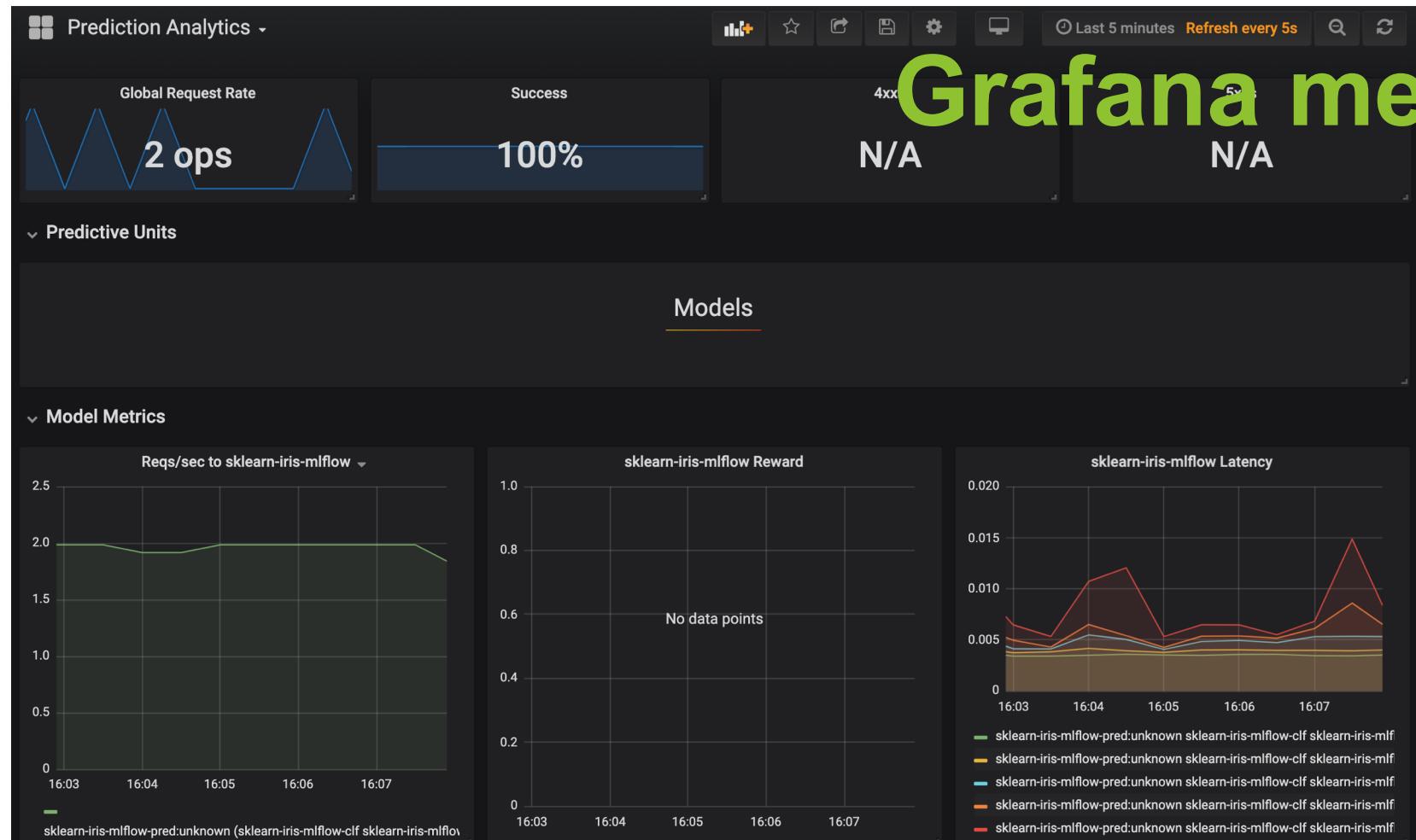
# Deploy the model

- Define the YAML / JSON Seldon deployment
- Build the image

```
s2i build -E environment_rest .  
seldonio/seldon-core-s2i-python3:0.6-SNAPSHOT  
sklearn-iris-mlflow:0.3
```

- Deploy

```
kubectl create -f sklearn_iris_deployment.json  
-n kubeflow
```



# COMCAST IS HIRING



PHILADELPHIA  
WASHINGTON, D.C.  
SILICON VALLEY  
DENVER



DON'T FORGET TO RATE  
AND REVIEW THE SESSIONS

SEARCH SPARK + AI SUMMIT

