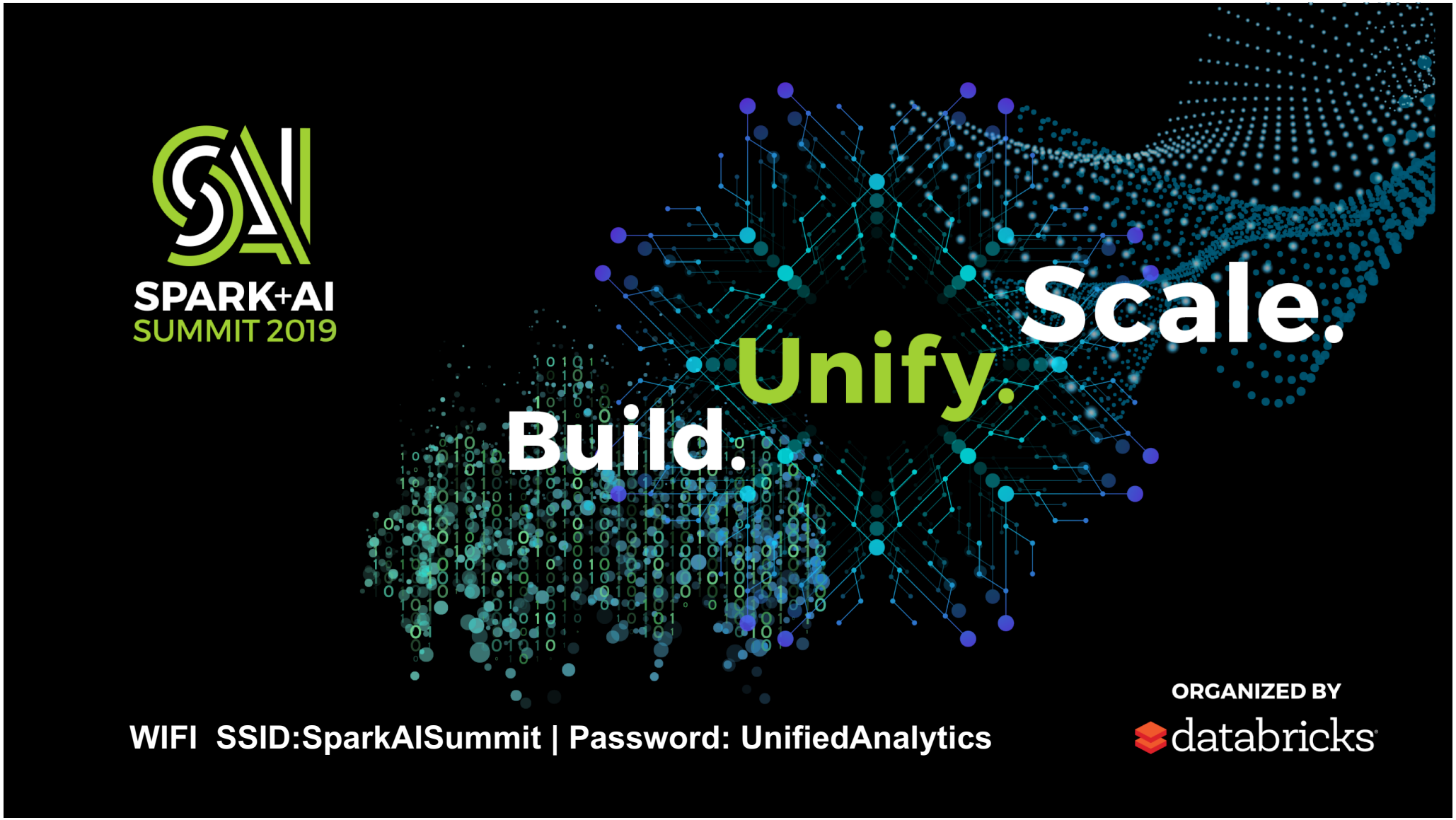
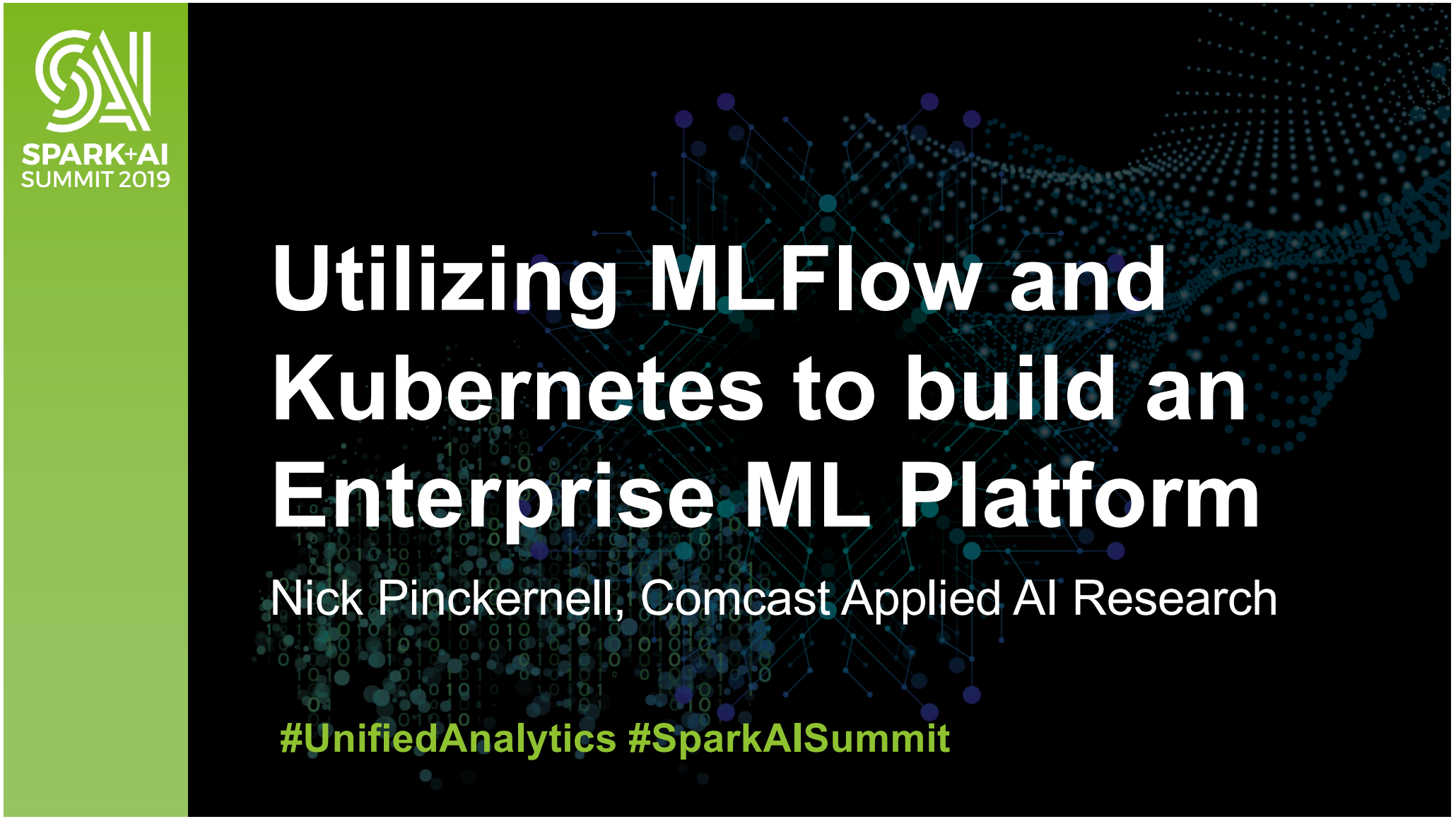

SPARK+AI
SUMMIT 2019
Build.
Unify.
Scale.
WIFI SSID:SparkAISummit | Password: UnifiedAnalytics
ORGANIZED BY
databricks



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

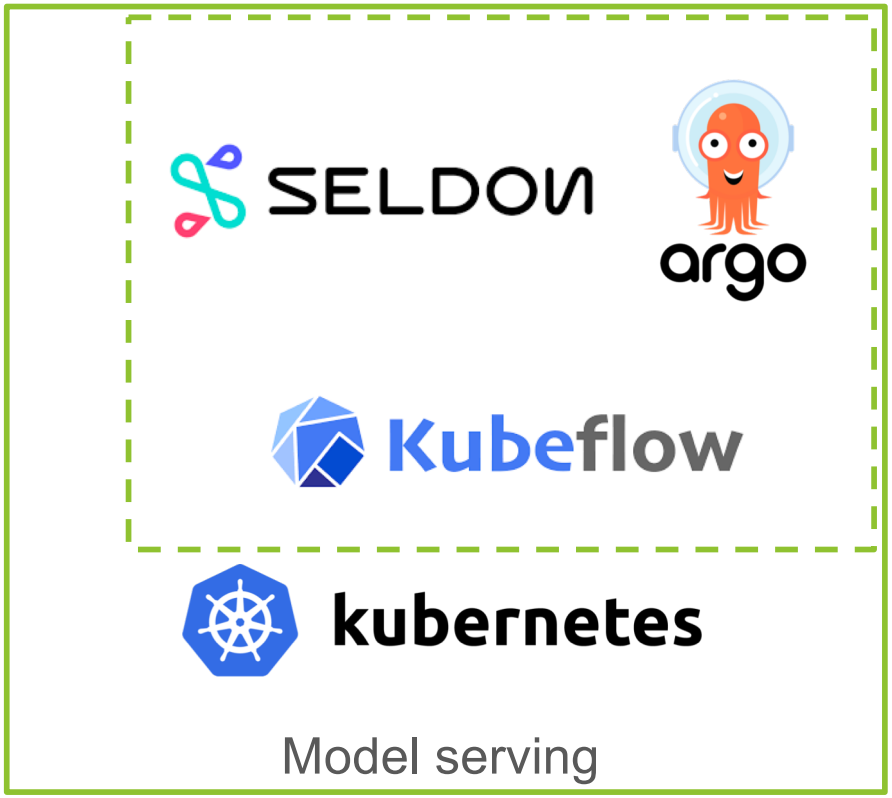
- Before, there was no
  - model management or tracking
  - standardization for model packaging or deployments
- Cumbersome deployment process
  - Deployment required code rewrite from research to operations
  - Days or weeks to deploy
- Response and tradeoff: restrict model complexity

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

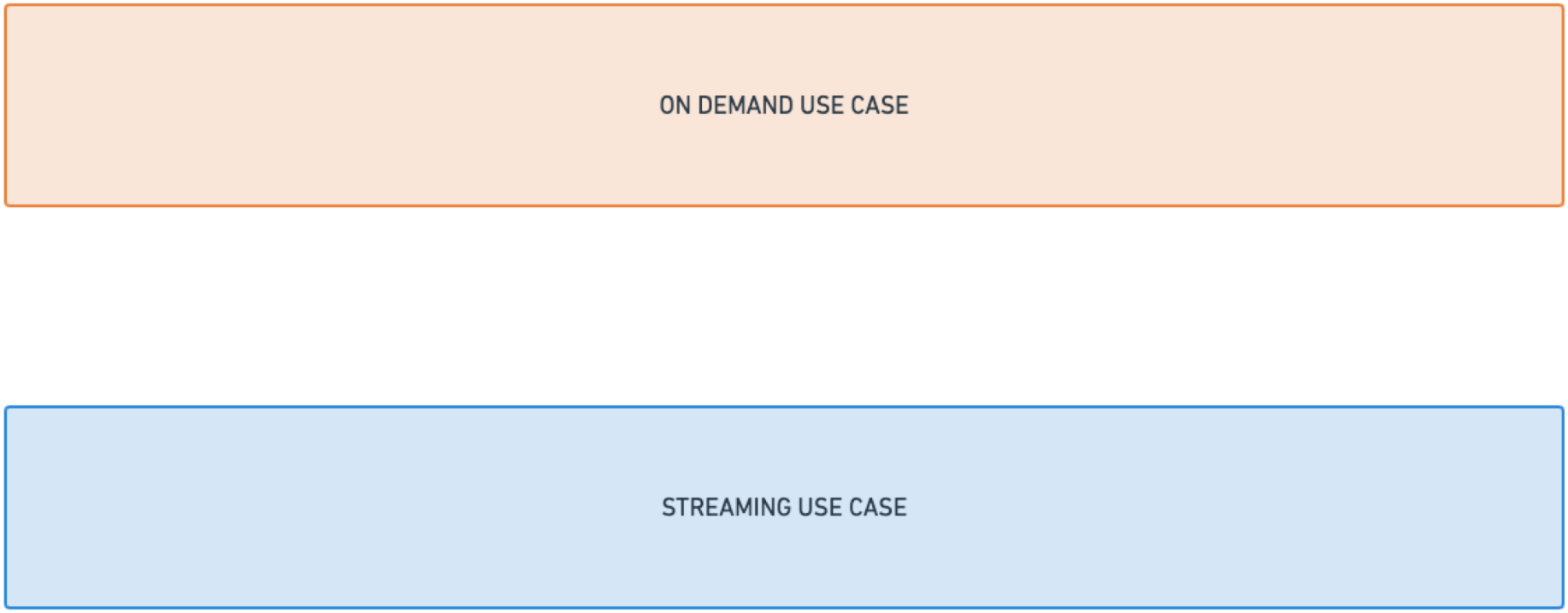
Solution – existing technologies



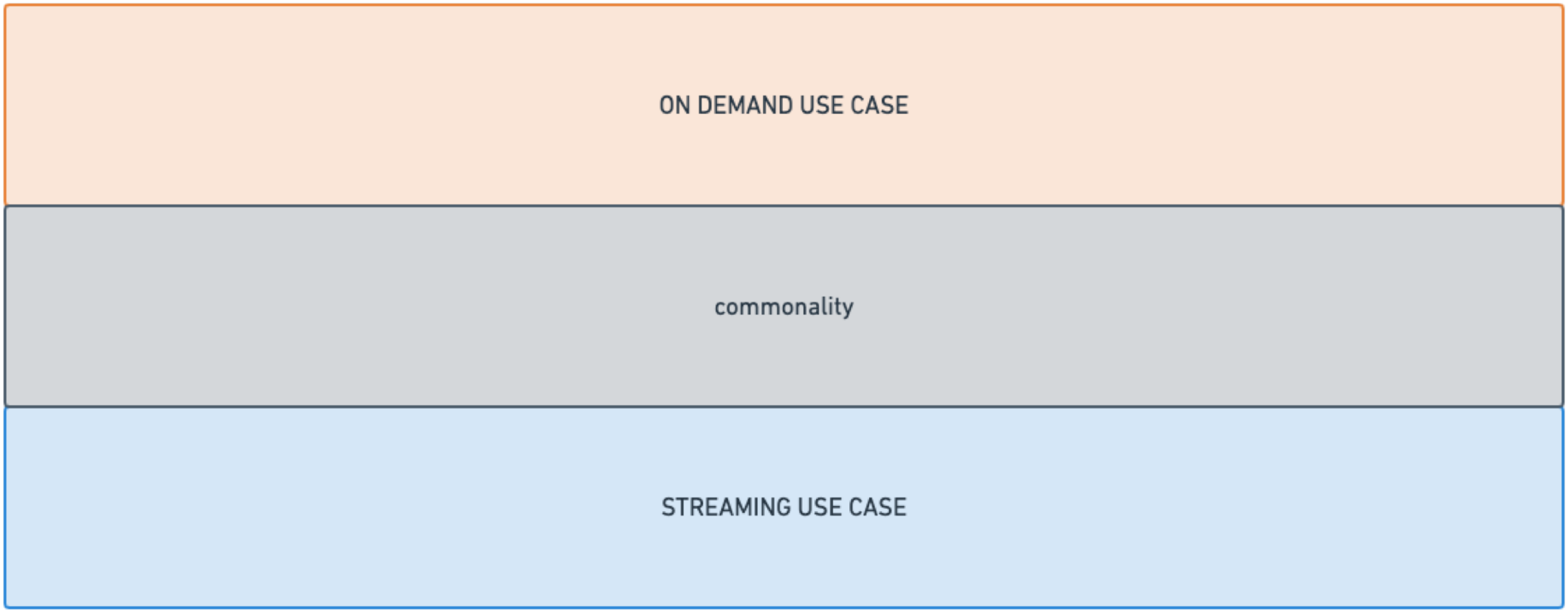
Data pipeline abstraction

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

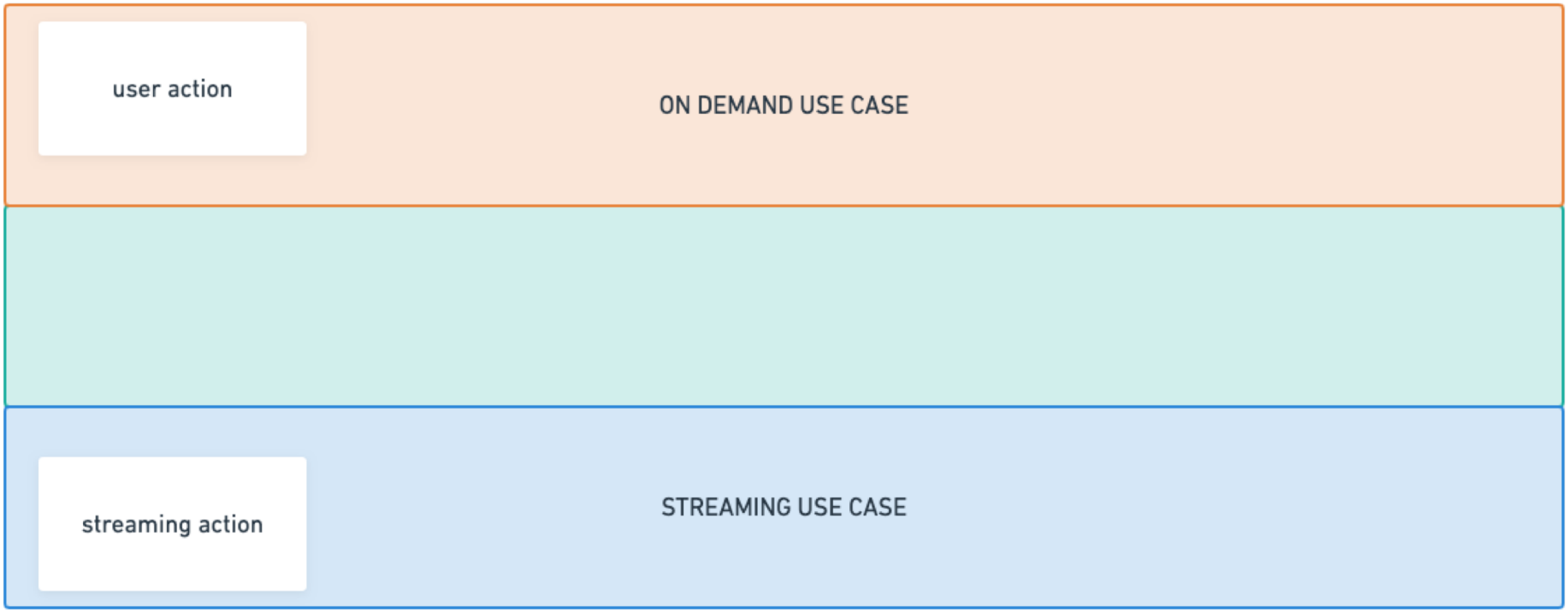
Pipeline abstraction



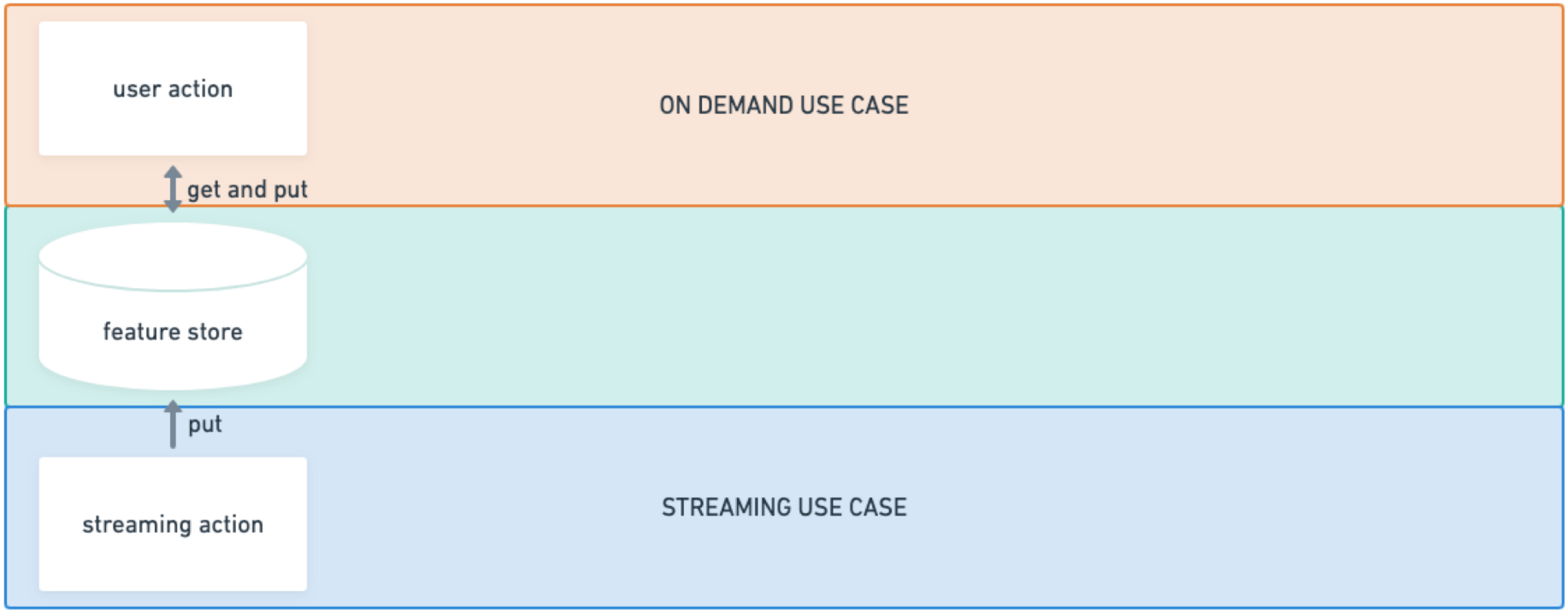
Pipeline abstraction



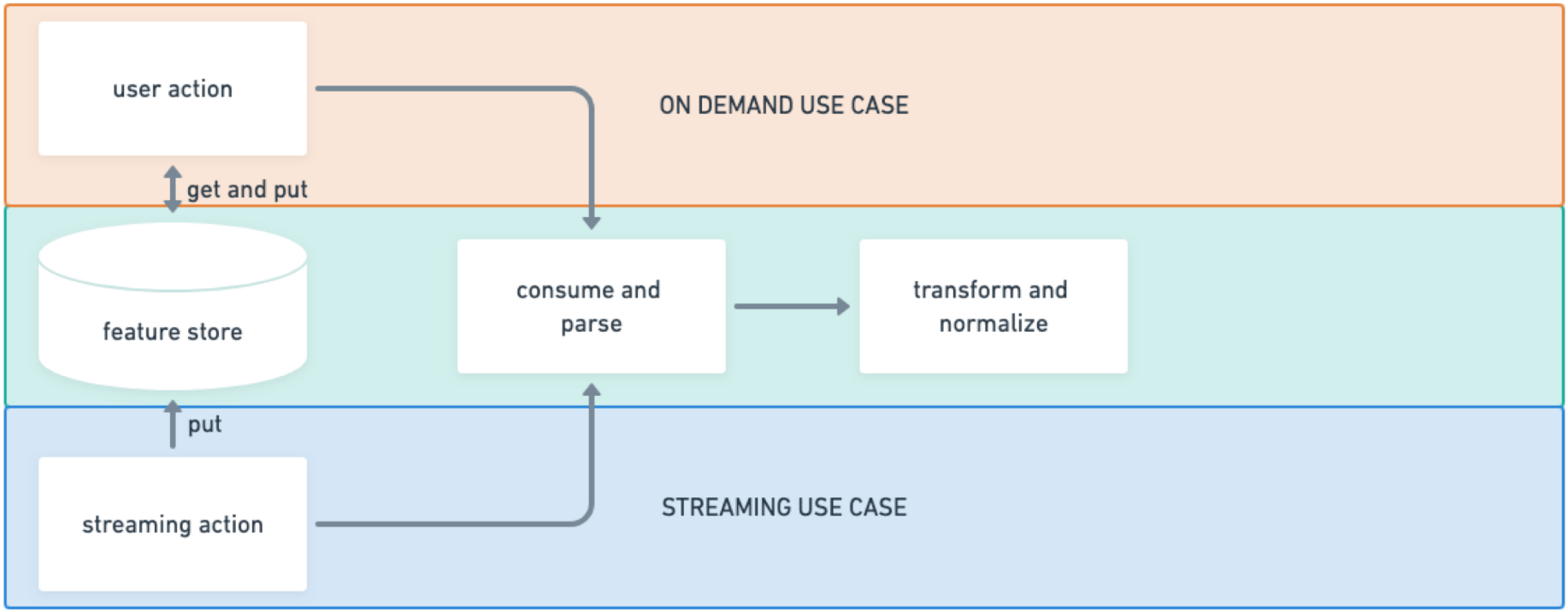
Pipeline abstraction



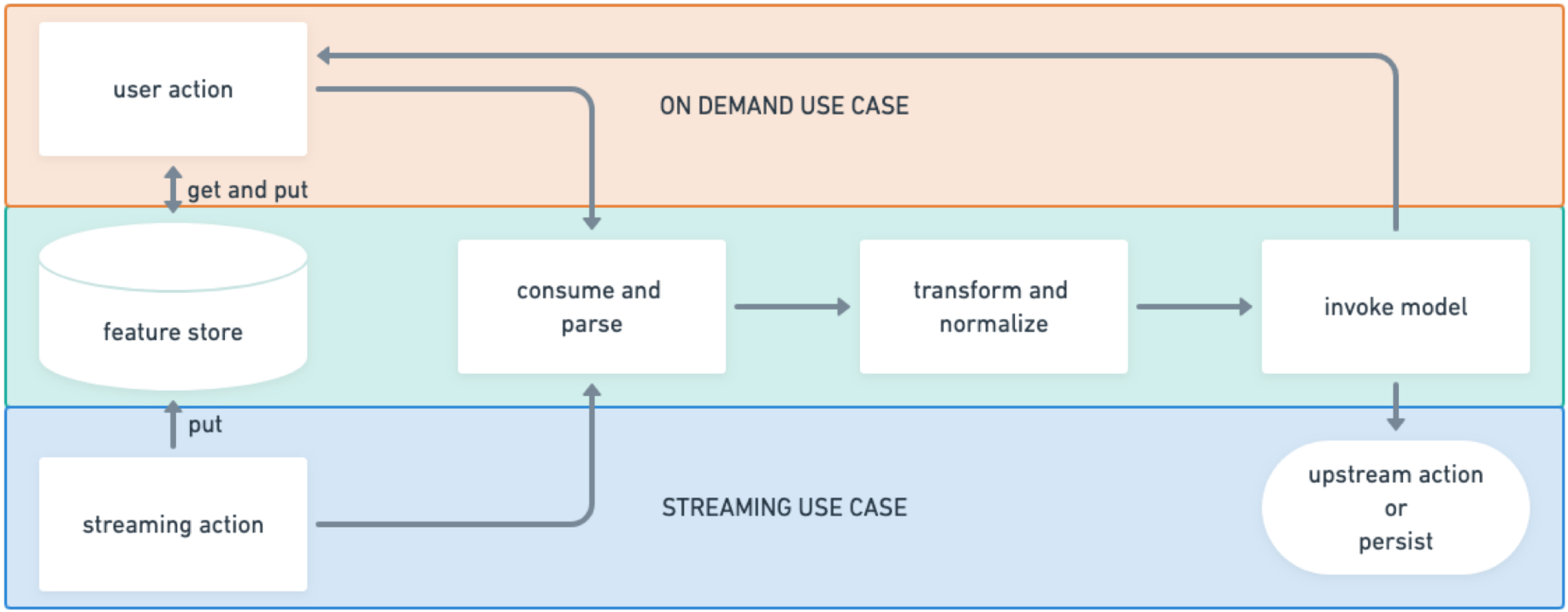
Pipeline abstraction



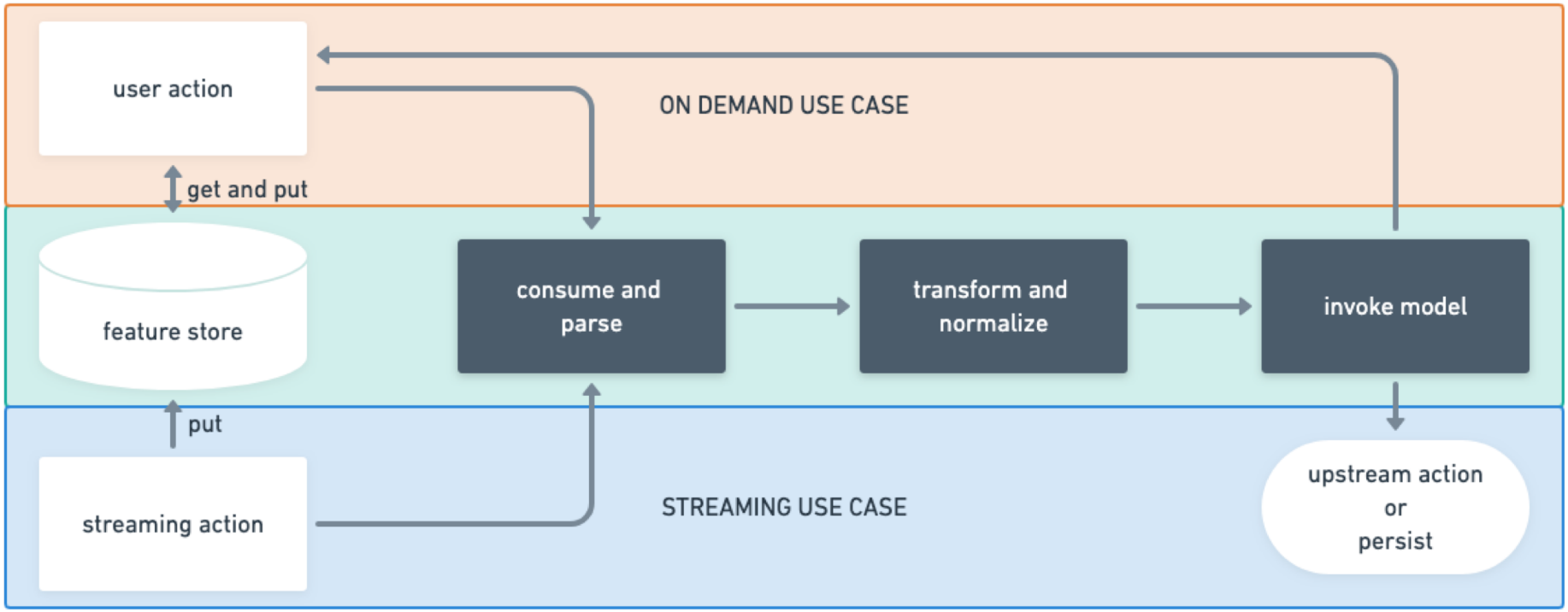
Pipeline abstraction



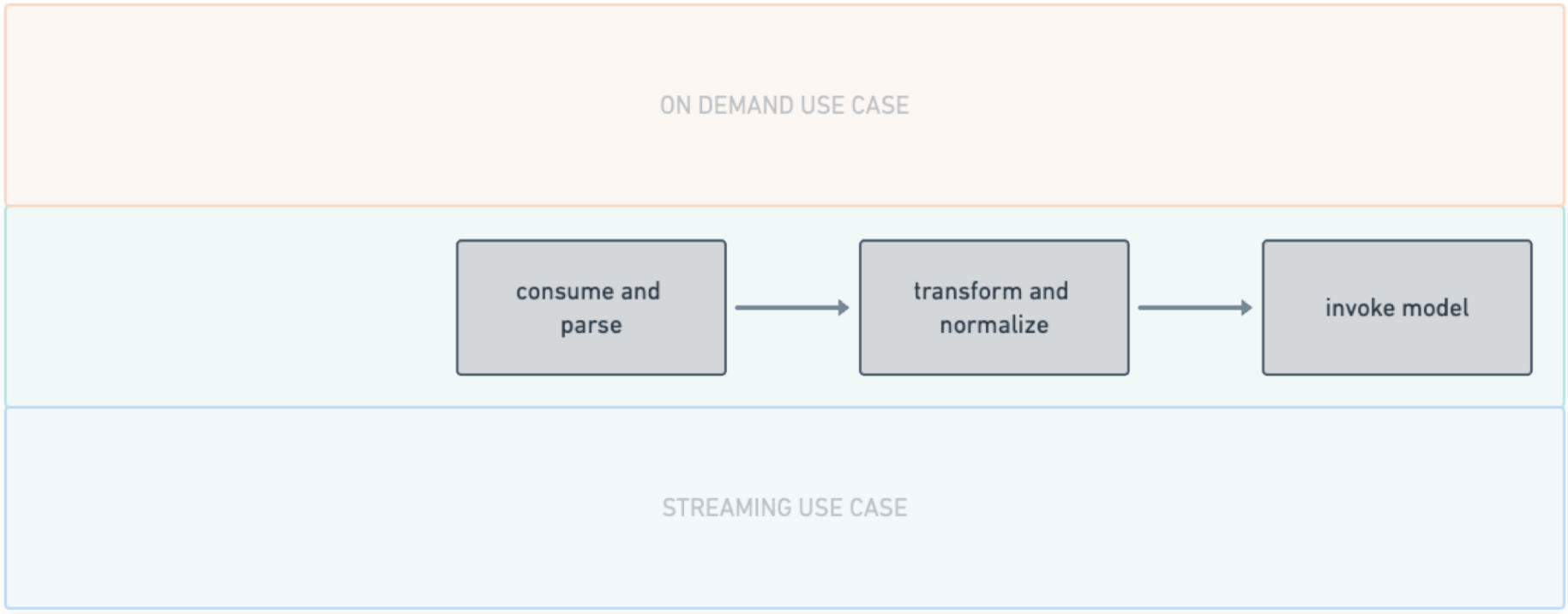
Pipeline abstraction



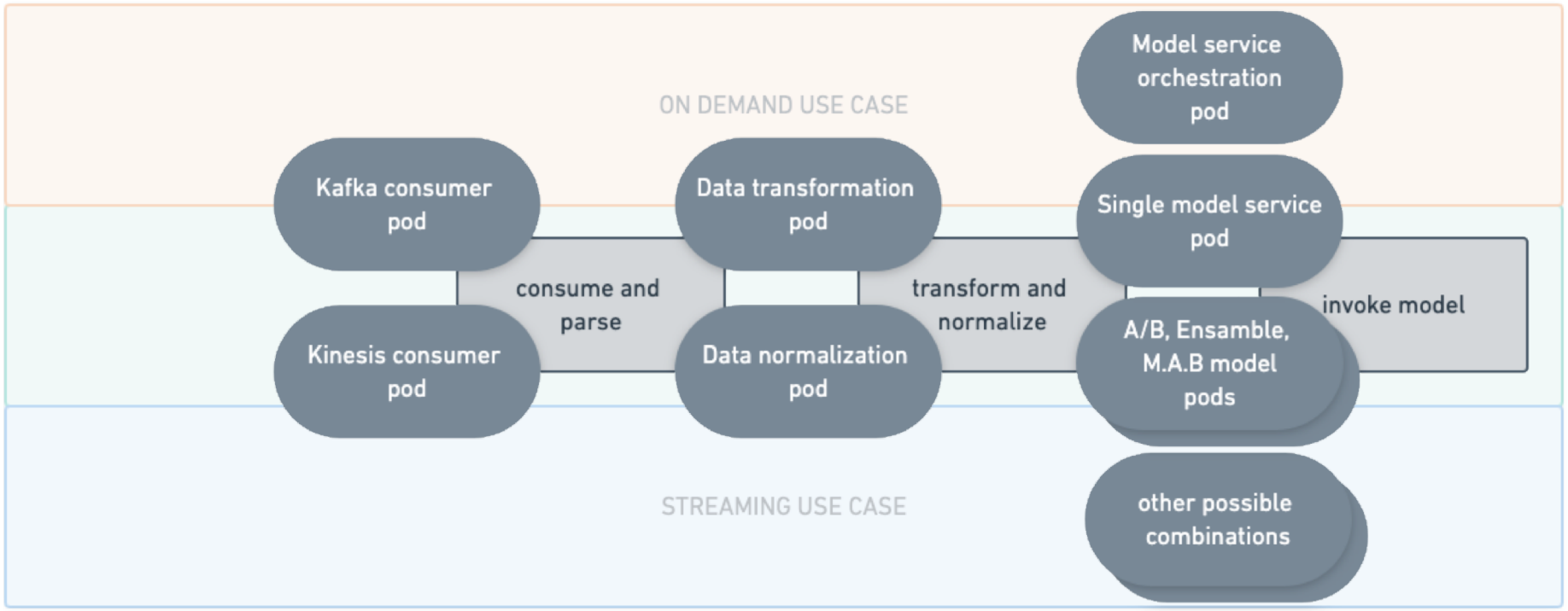
Pipeline abstraction



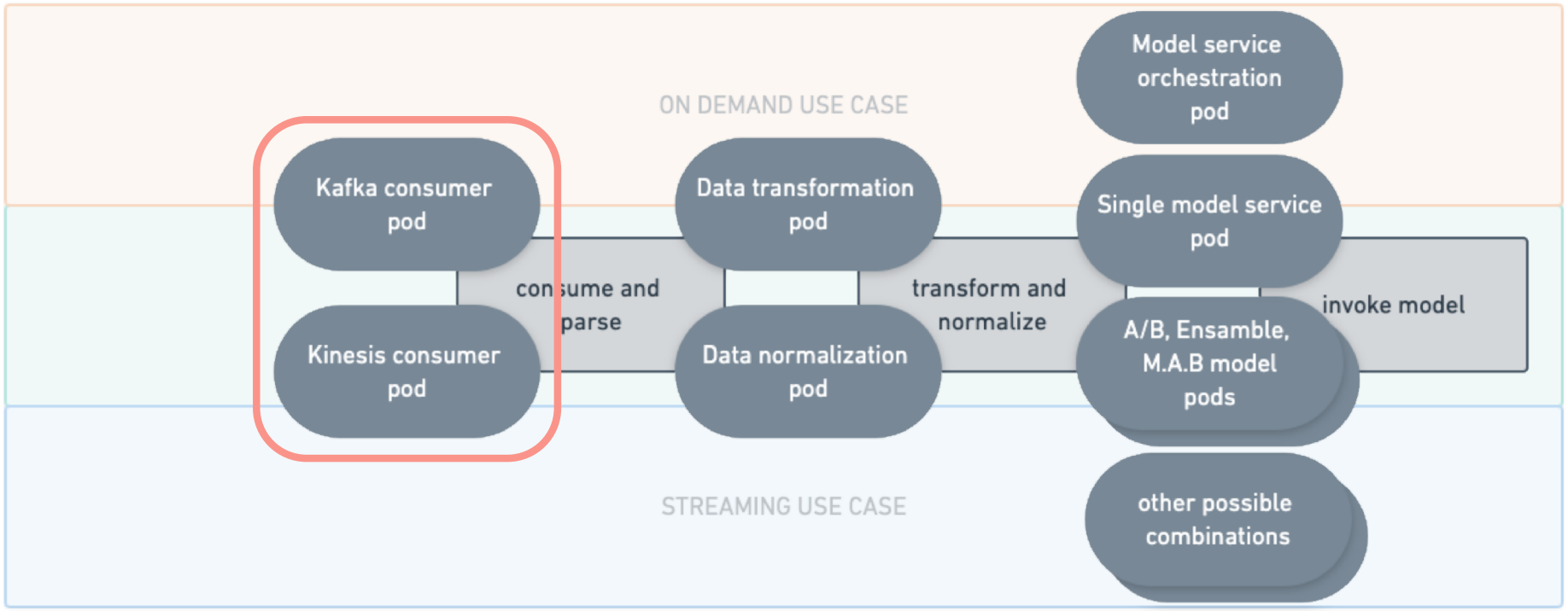
Pipeline abstraction



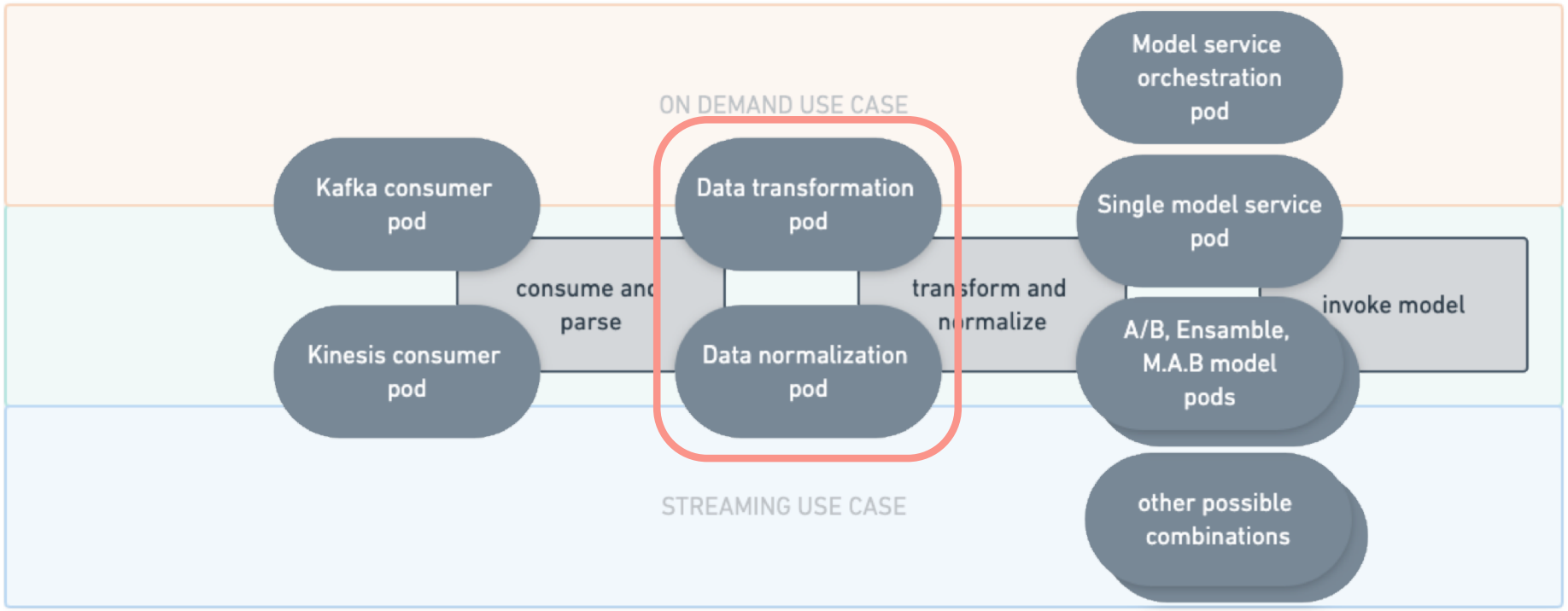
Pipeline abstraction



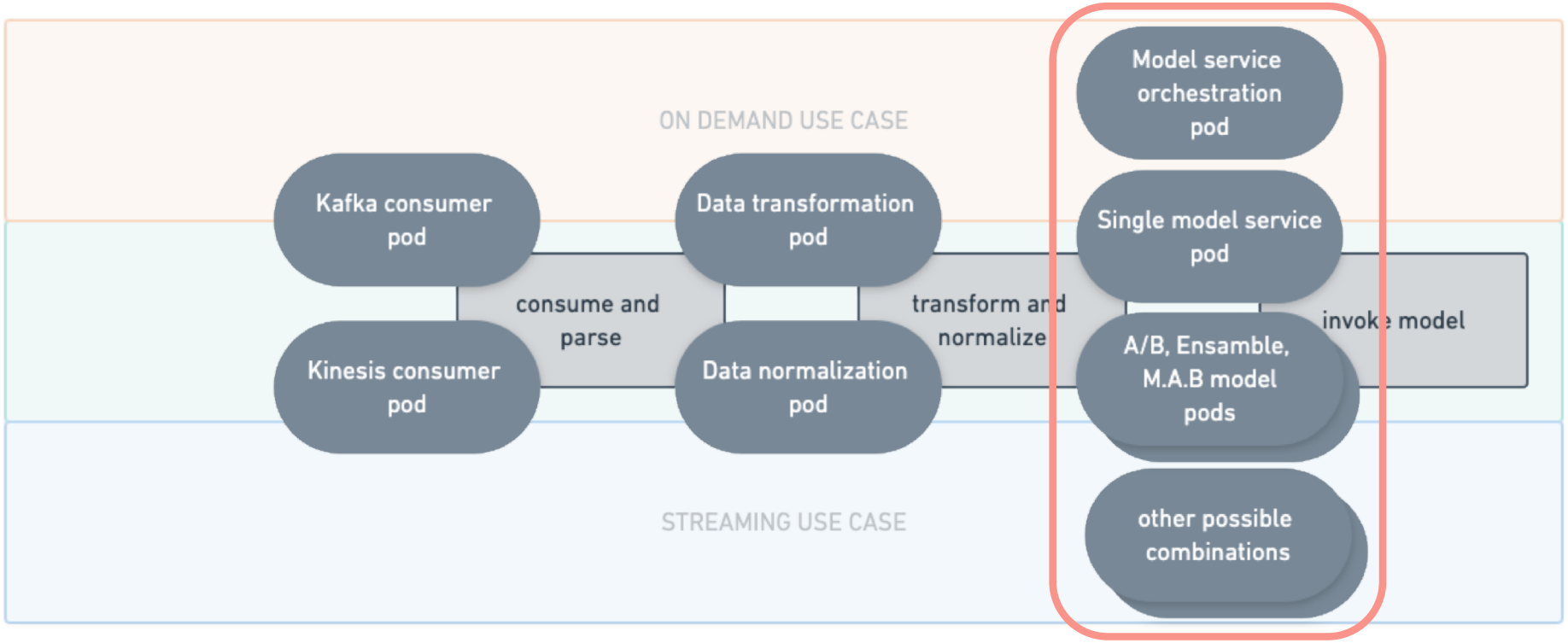
Pipeline abstraction



Pipeline abstraction



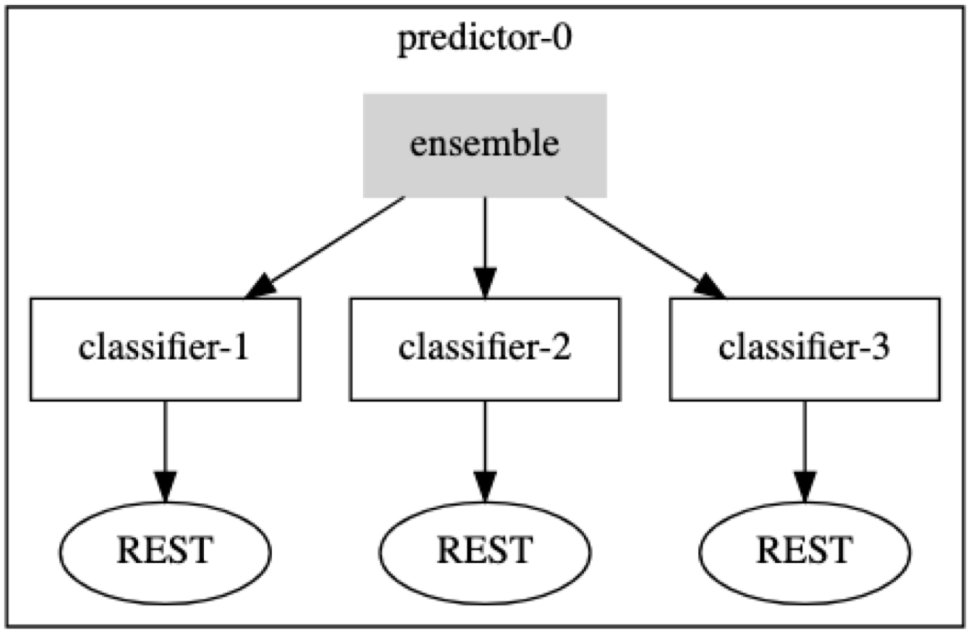
Pipeline abstraction



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

Packaging and tracking

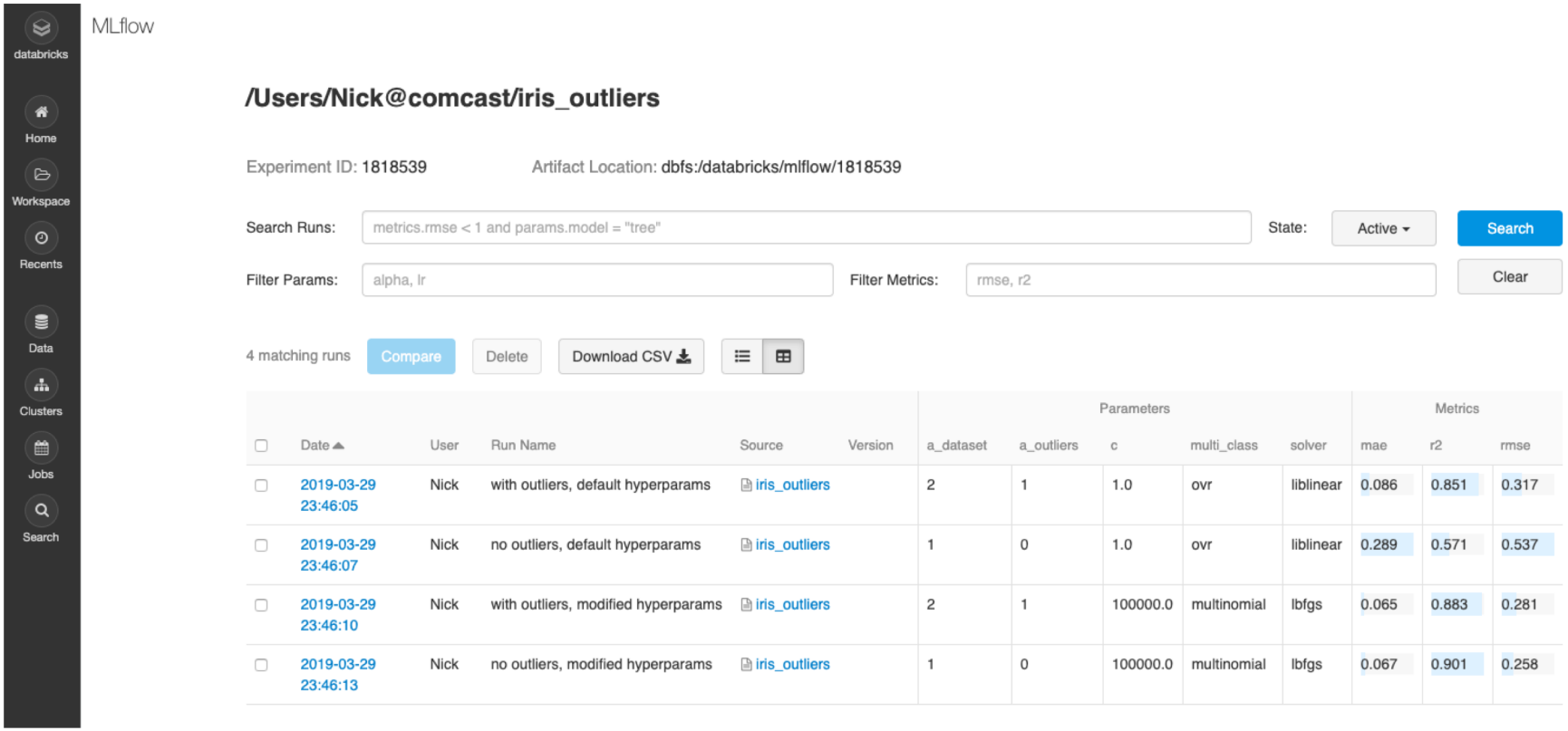
1. Researchers code and train models with Databricks, Spark
2. Experiments tracked with MLFlow
3. Packaging and model tracking with MLFlow and Kubeflow

- MLFLow standard packaging formats
  - scikit-learn
  - h2o
  - TensorFlow
  - more

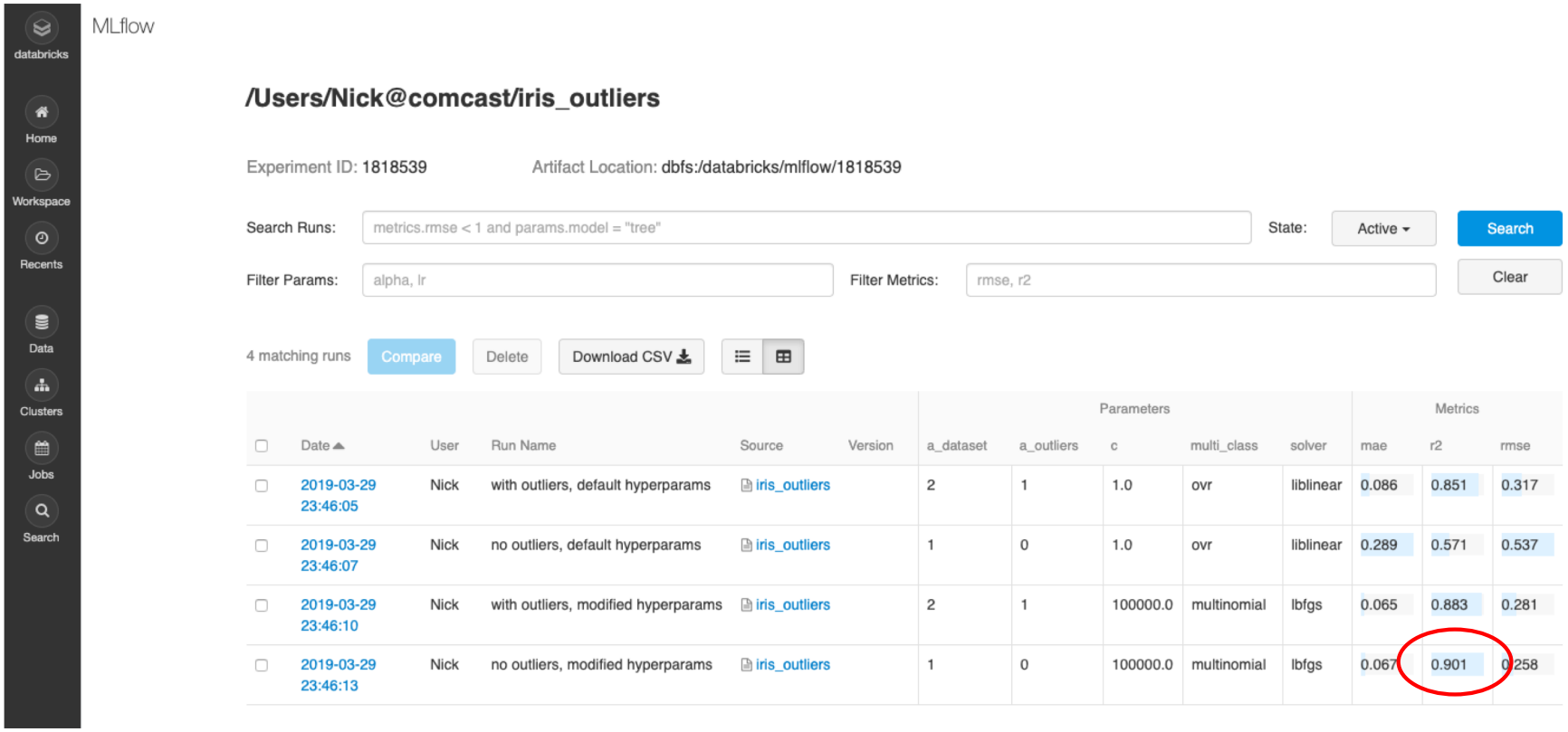
An MLFlow experiment

```
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

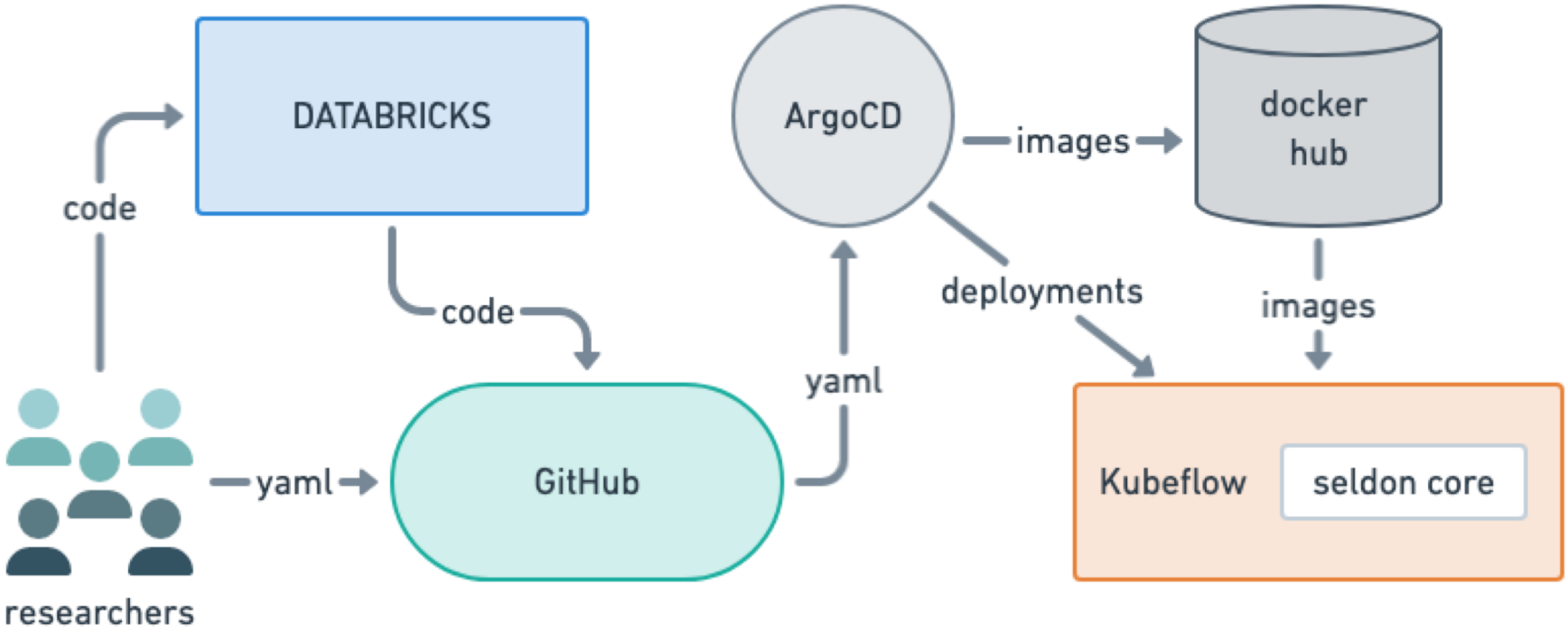


MLFlow – multiple experiments

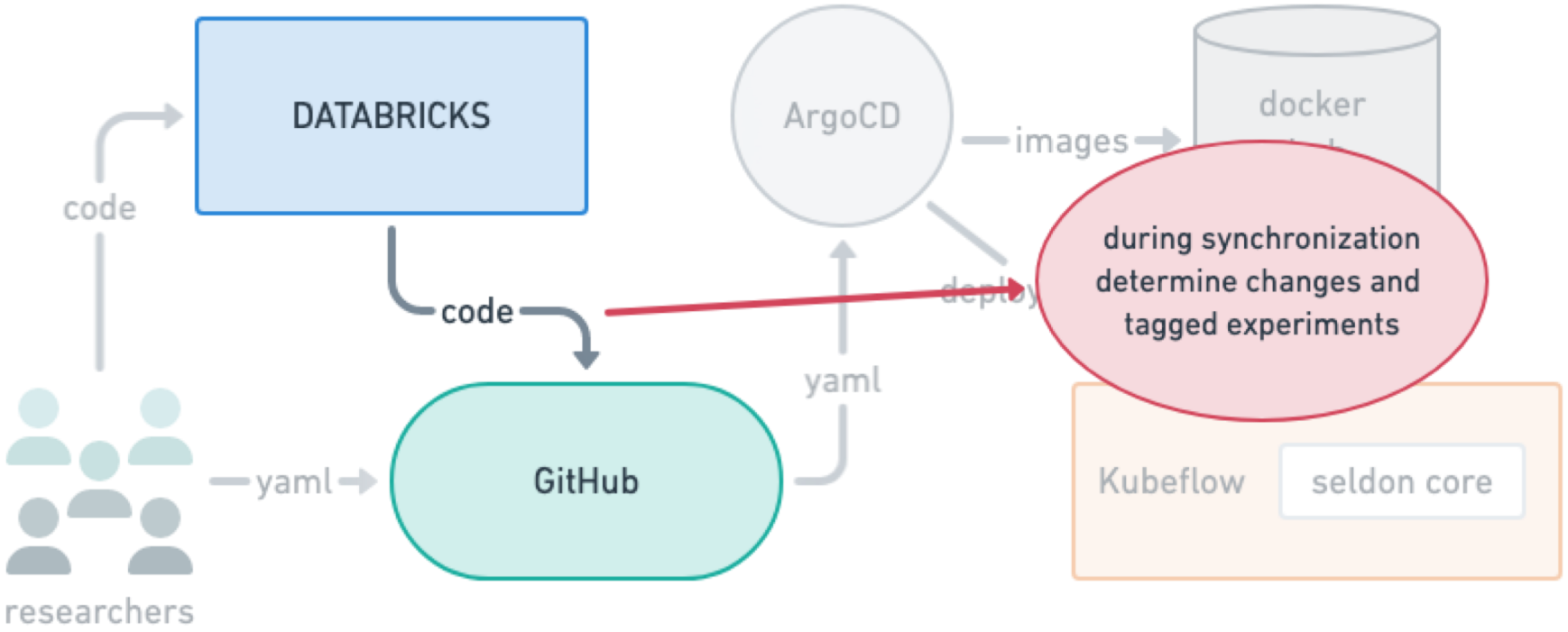


MLFlow packaging

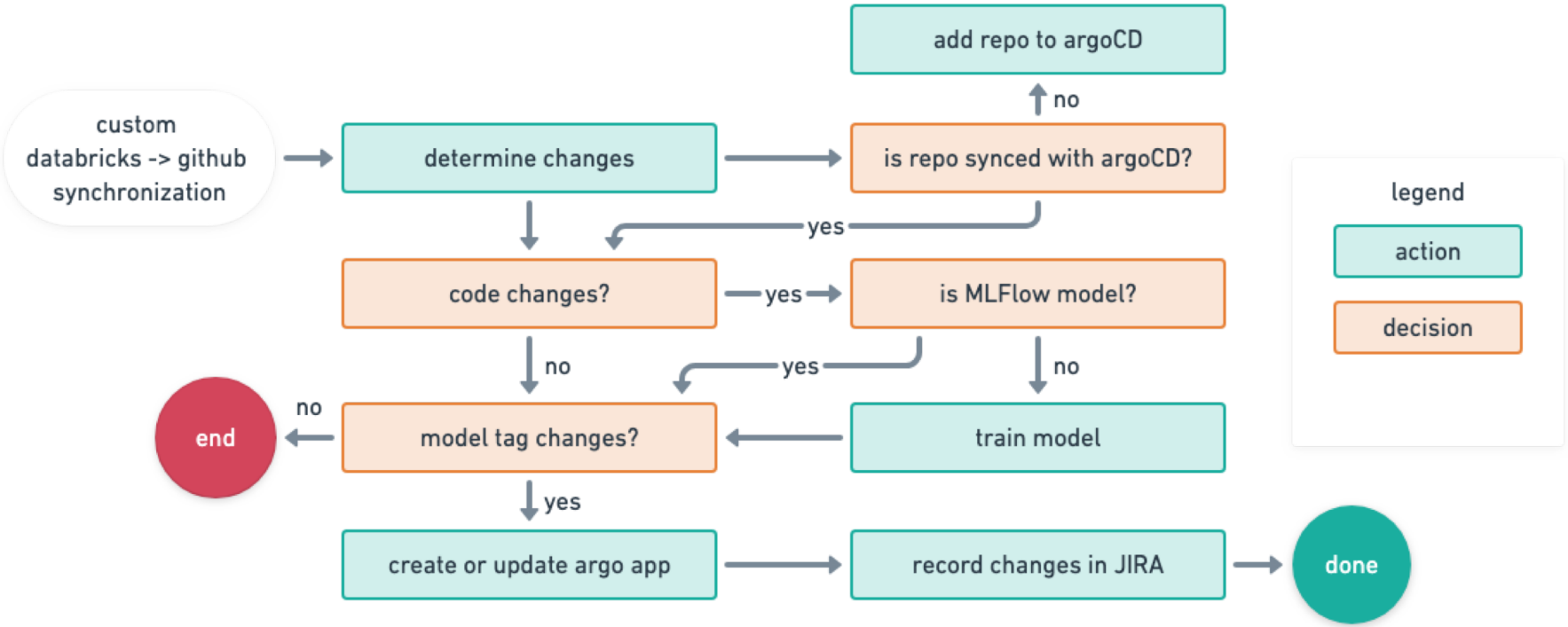
Research and model flow



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)

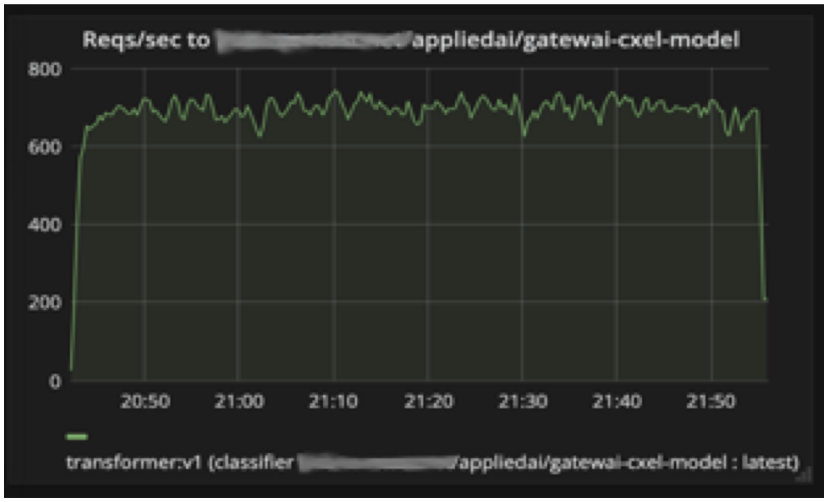
Ambassador

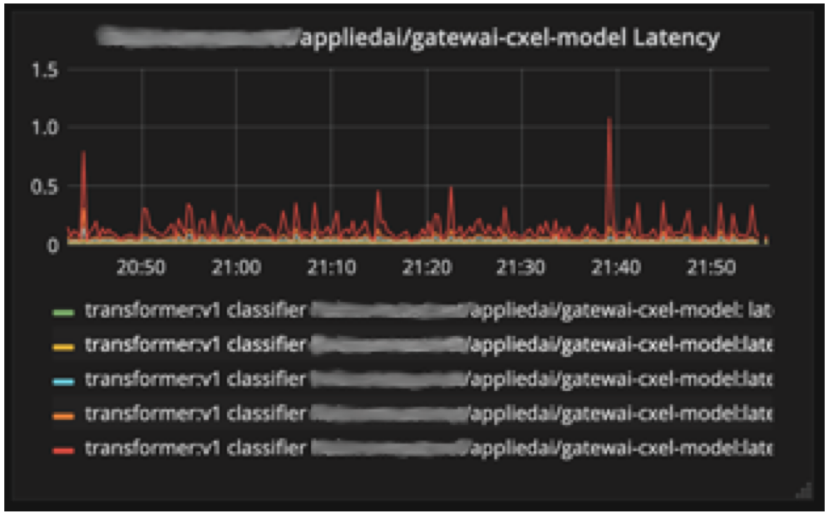
http://www.getambassador.io

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

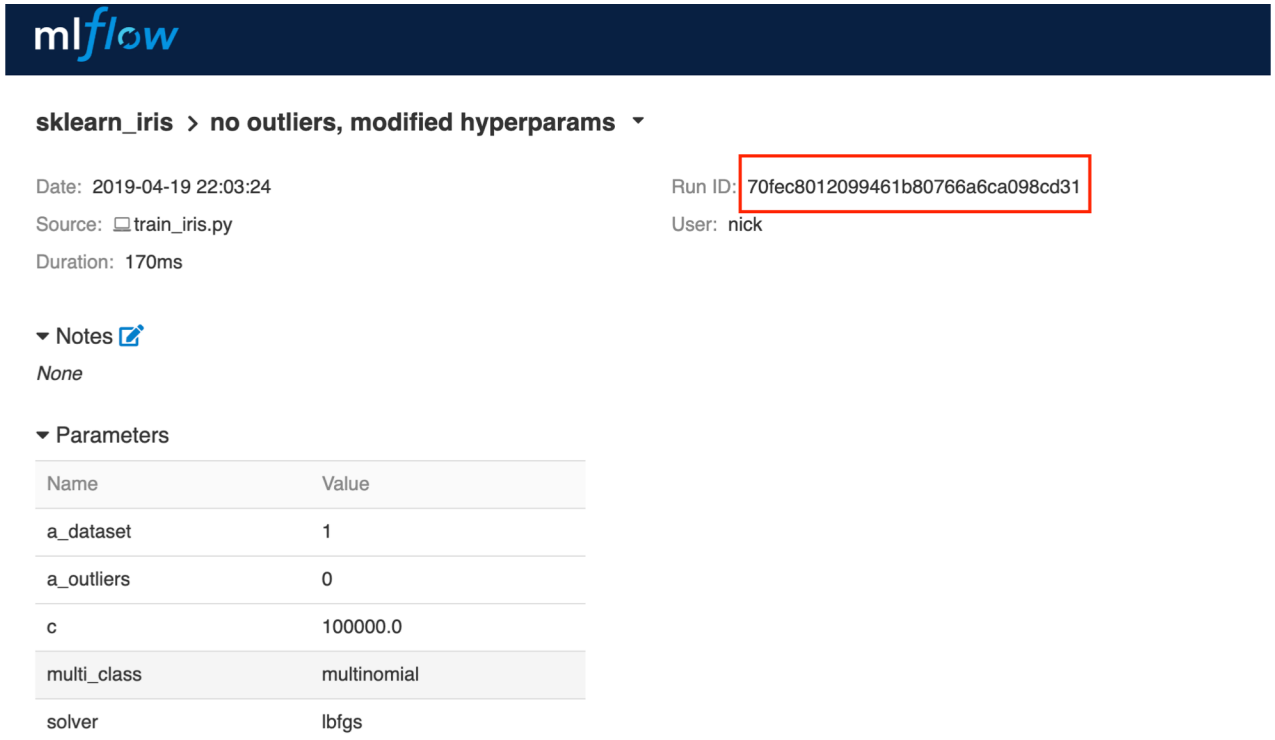
Search Runs: metrics.rmse < 1 and params.model = "tree" State: Active Search

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

4 matching runs Compare Delete Download CSV

						Parameters					Metrics			
	Date	User	Run Name	Source	Version	a_dataset	a_outliers	c	multi_class	solver	acc	mae	r2	rmse
	2019-04-19 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 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 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 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



Implementing the model

```
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 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

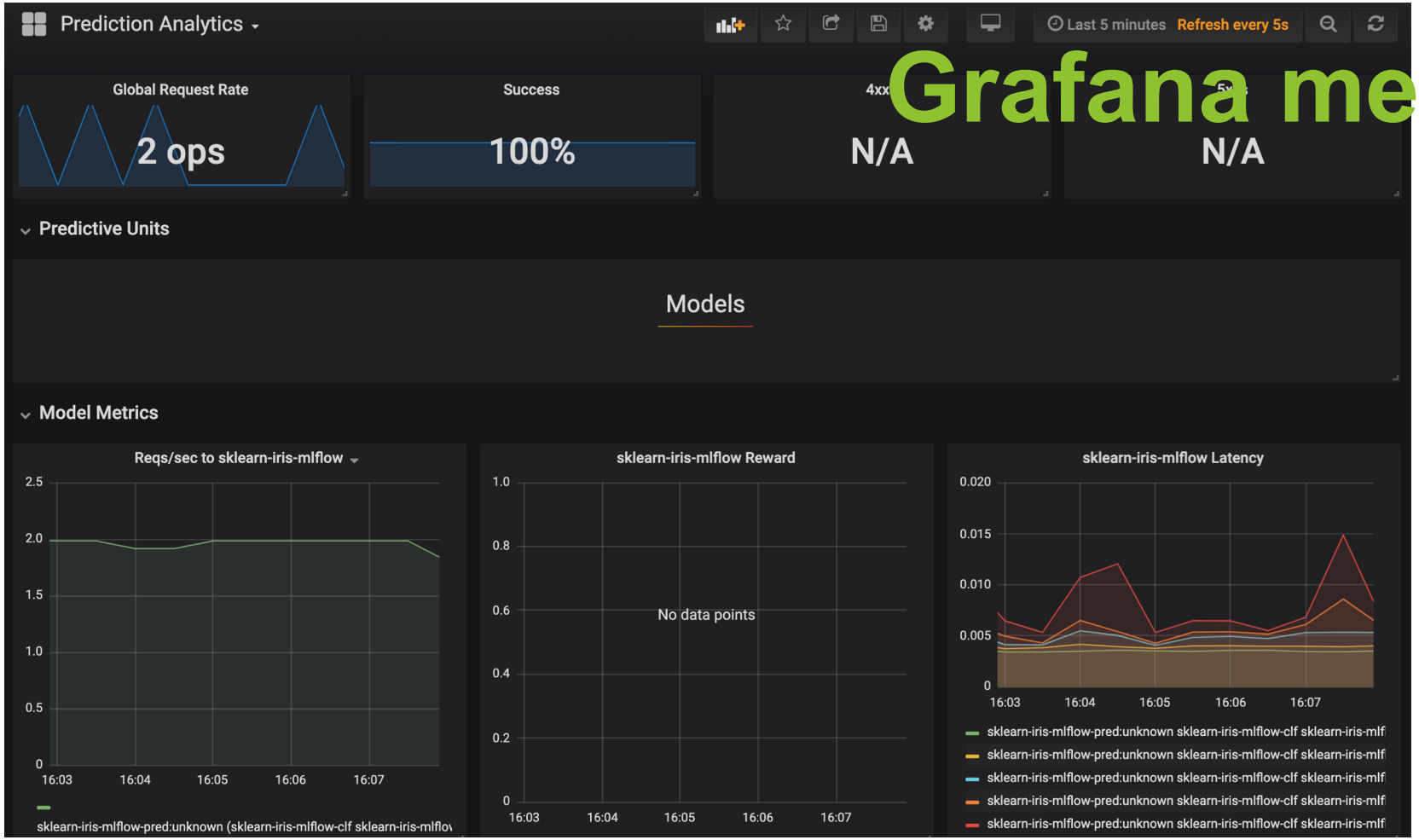
- Define the YAML / JSON Seldon deployment
- Build the image

```
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
```

Grafana metrics



COMCAST IS HIRING

PHILADELPHIA

WASHINGTON, D.C.

SILICON VALLEY

DENVER

SPARK+AI SUMMIT 2019

DON'T FORGET TO RATE AND REVIEW THE SESSIONS

SEARCH SPARK + AI SUMMIT

Download on the App Store

GET IT ON Google Play