WORKSHOP DAY ONSITE
OCTOBER 4, 2023

CONFERENCE DAY ONSITE
OCTOBER 5, 2023

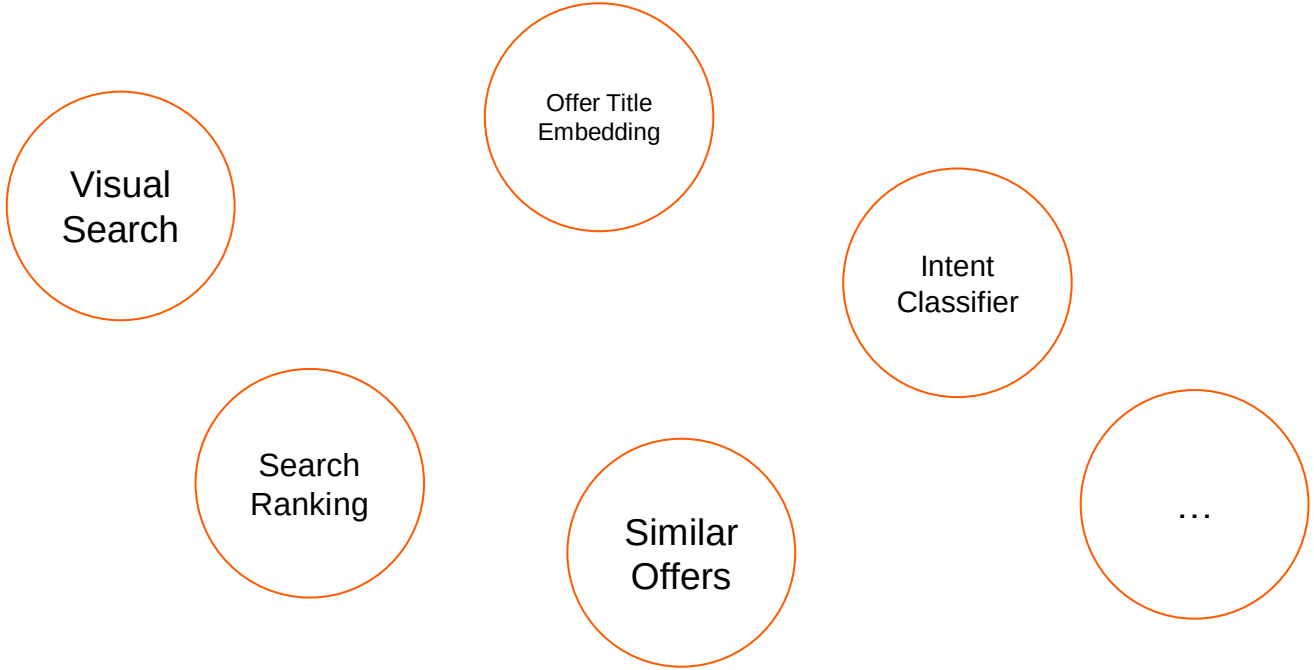
WORKSHOP DAY ONLINE
OCTOBER 6, 2023

Serving ML Models at Scale at Allegro

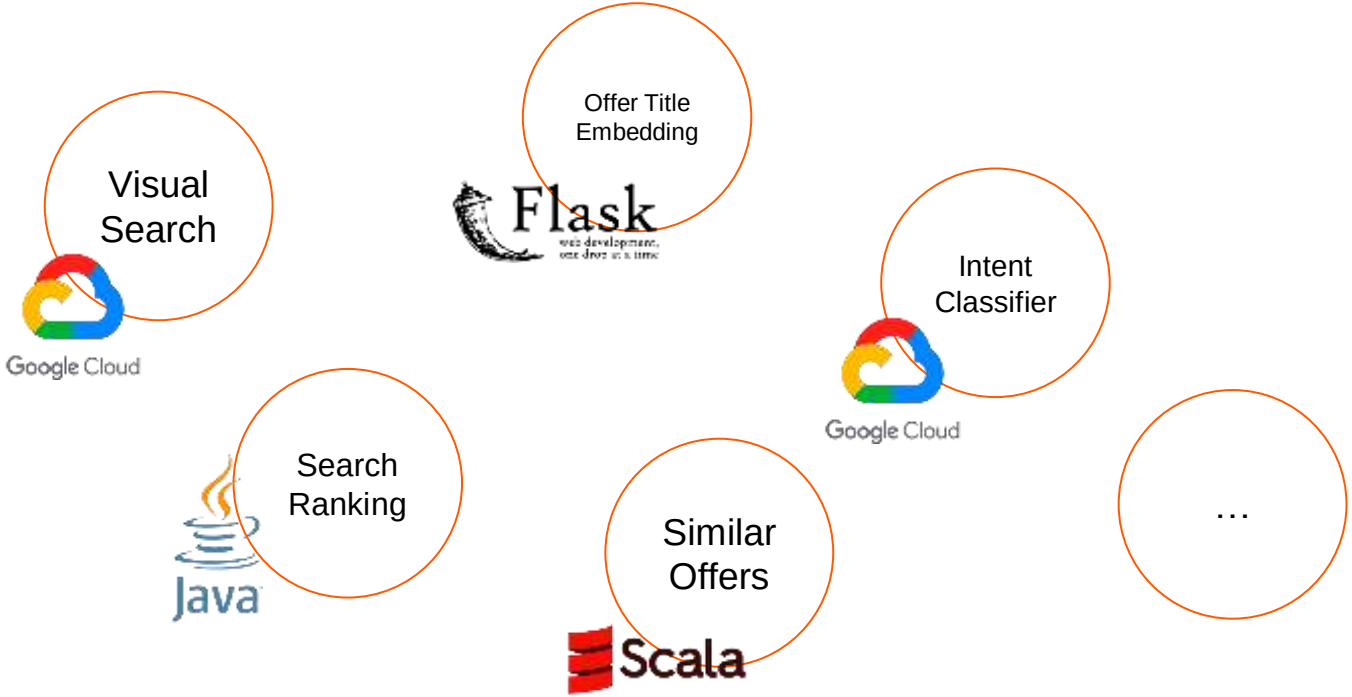
Agnieszka Rybak

SUMMIT.DATAMASS.IO

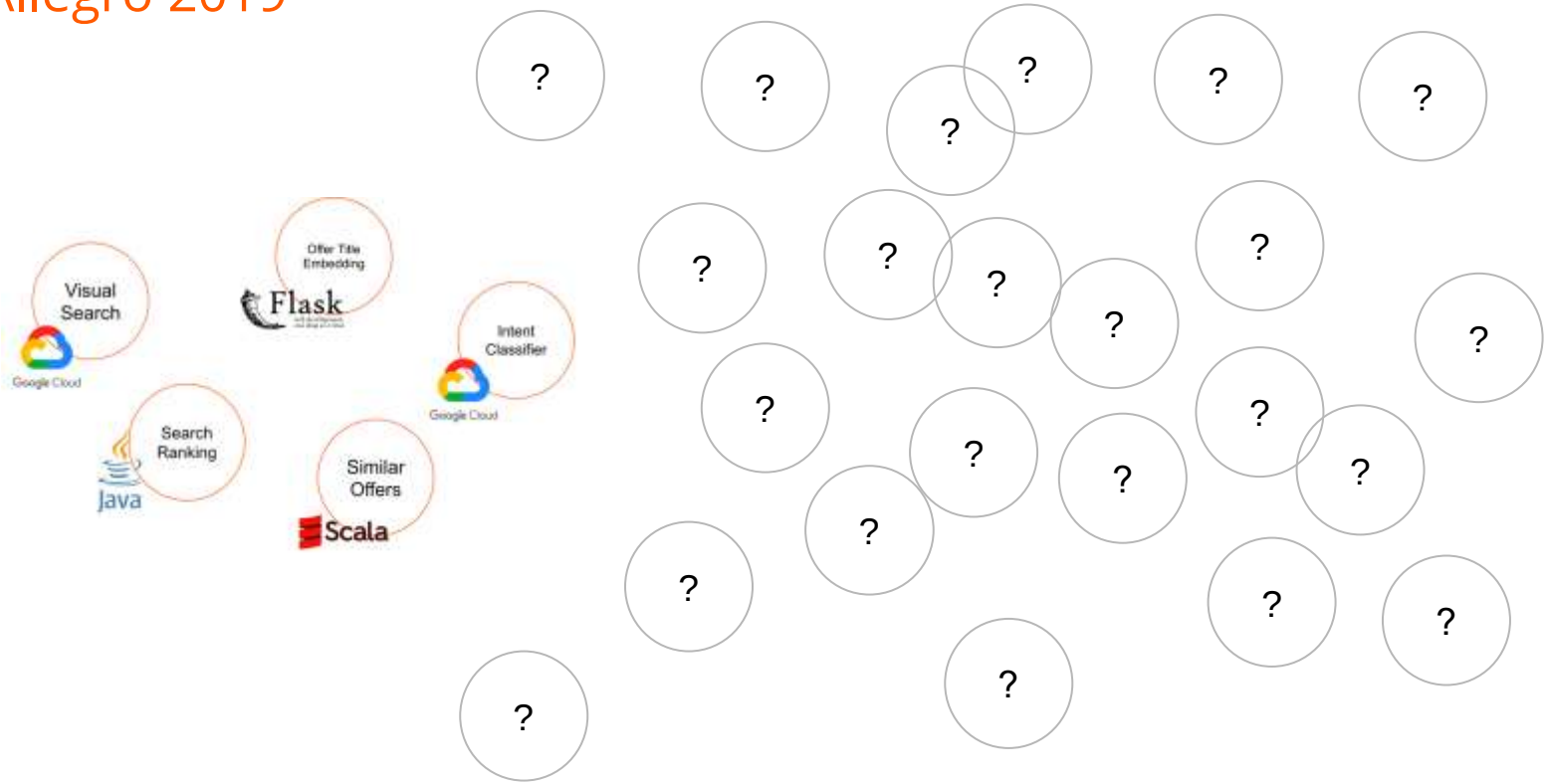
Allegro 2019



Allegro 2019



Allegro 2019



Allegro 2019

Robust, fast, good DevOps practices

Developed by software engineers

Time consuming to implement and deploy

Easy to implement by Researchers

Beta at the time

High latency, unstable

What we had

PagerDuty

Grafana

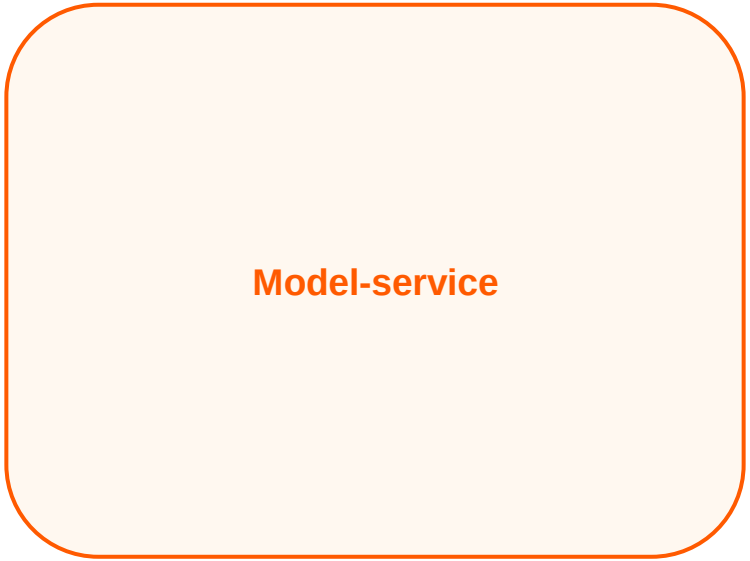
GitHub Actions

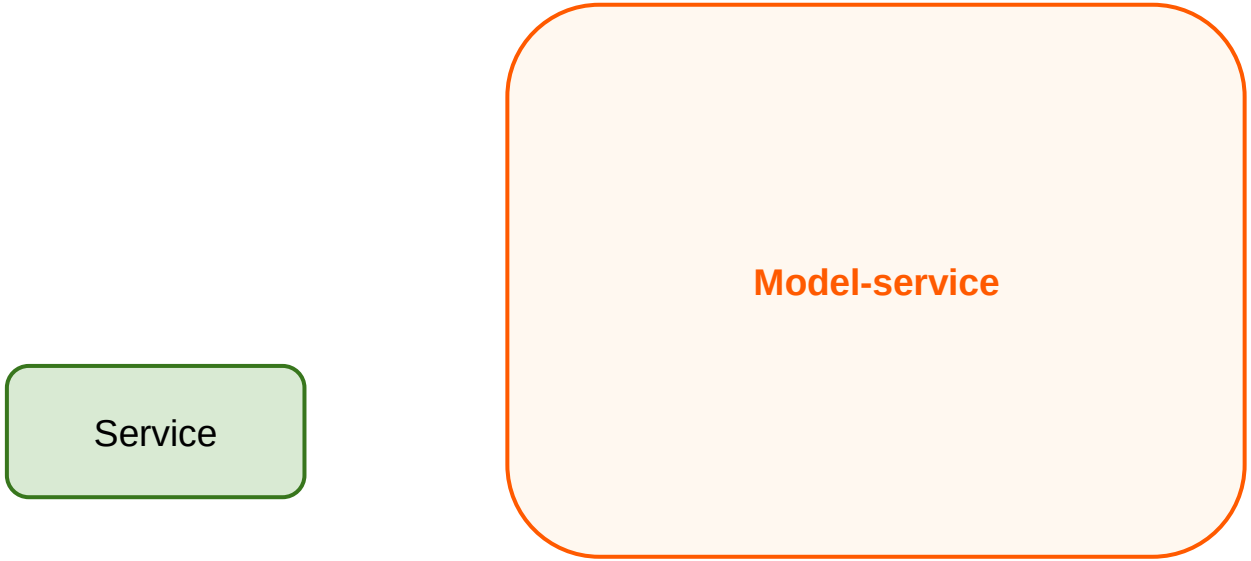
kubernetes

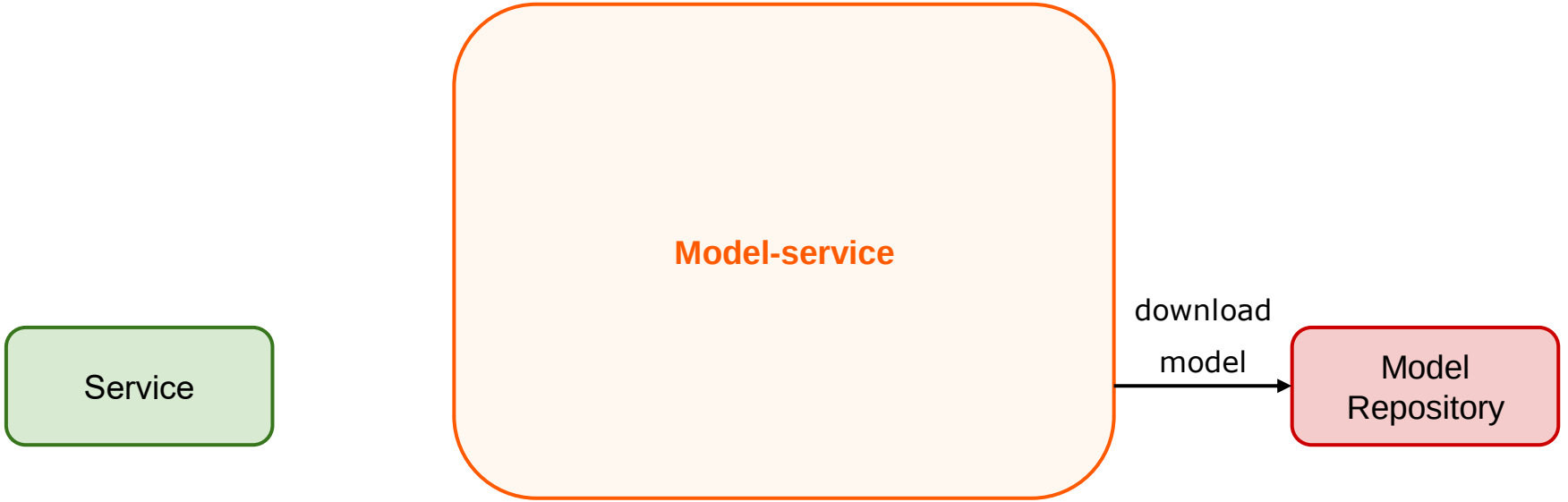
Kibana

Prometheus

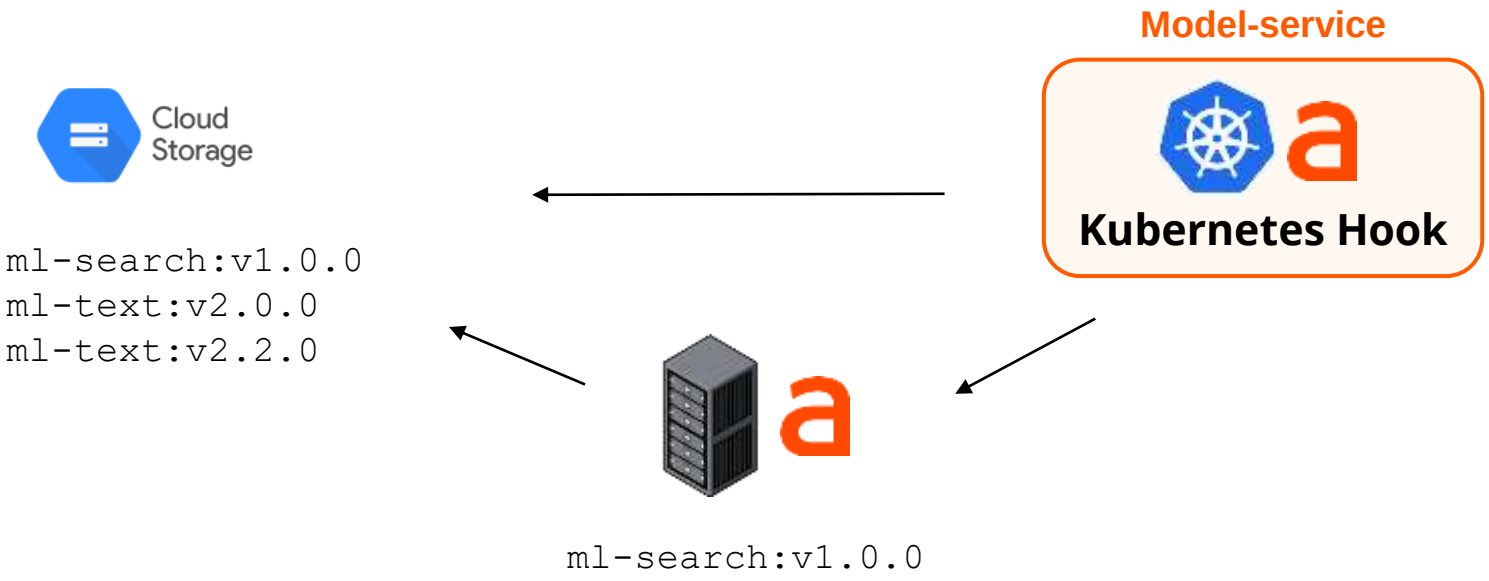
ML Platform based on App Engine


Model-service

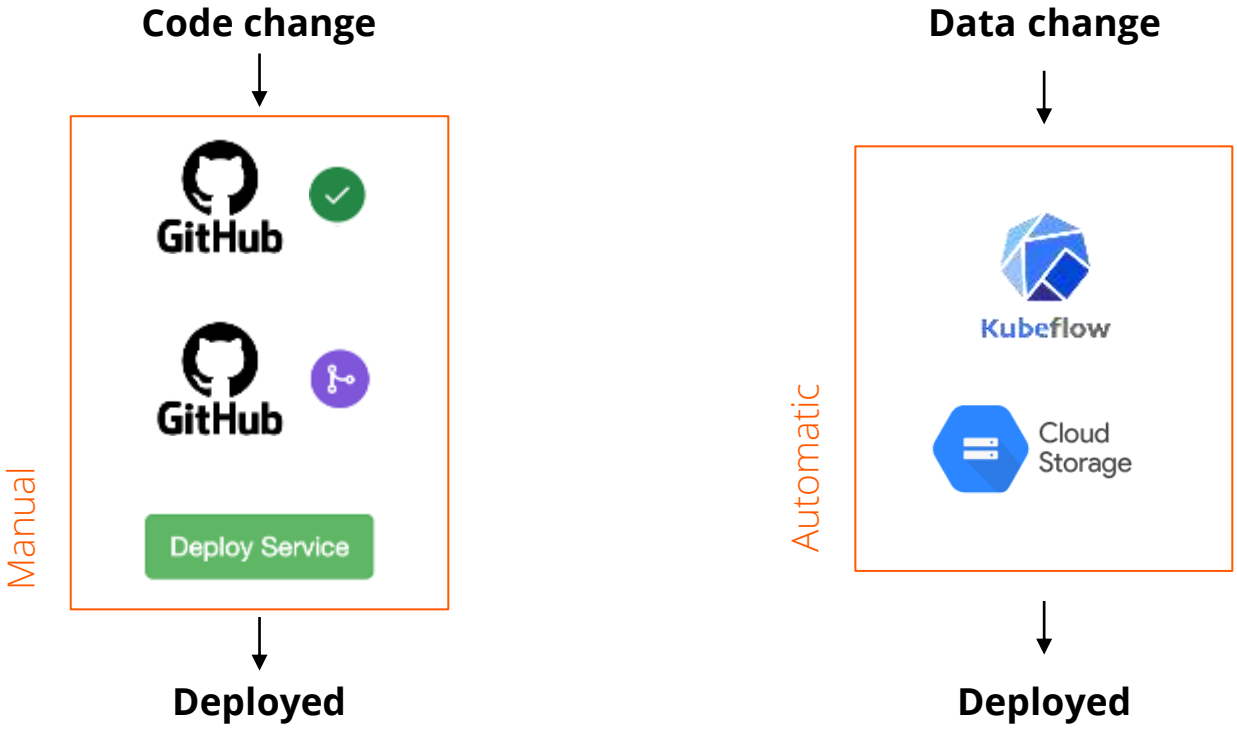

Model-service
Service

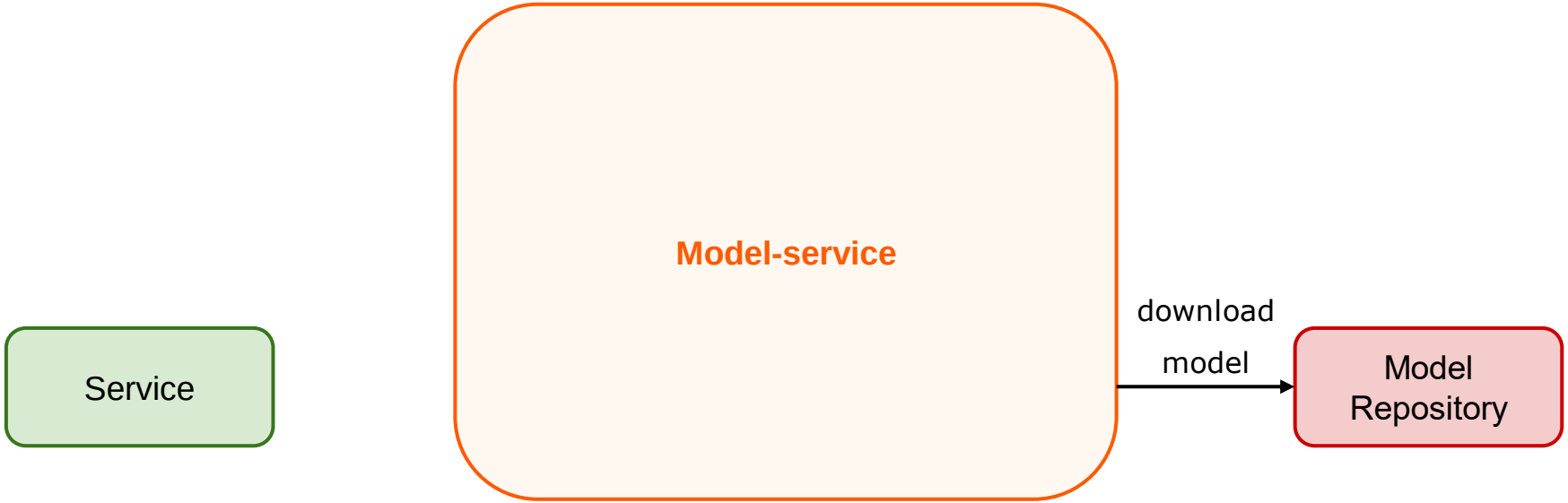

Model-service
Service
download
model
Model
Repository

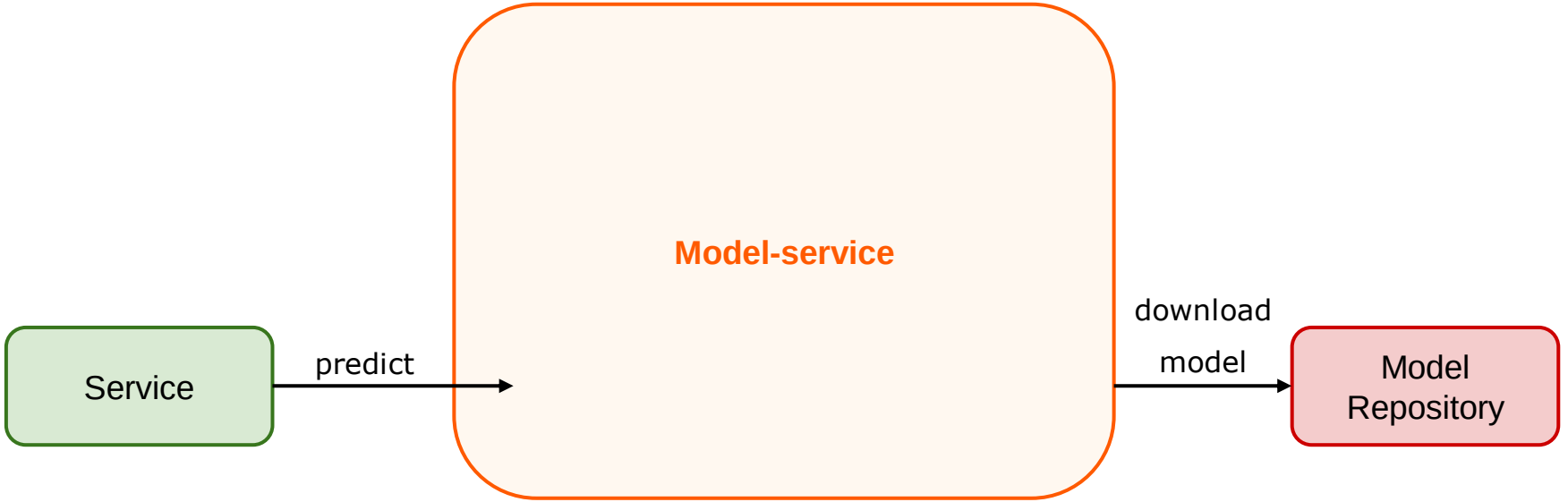
Model Repository

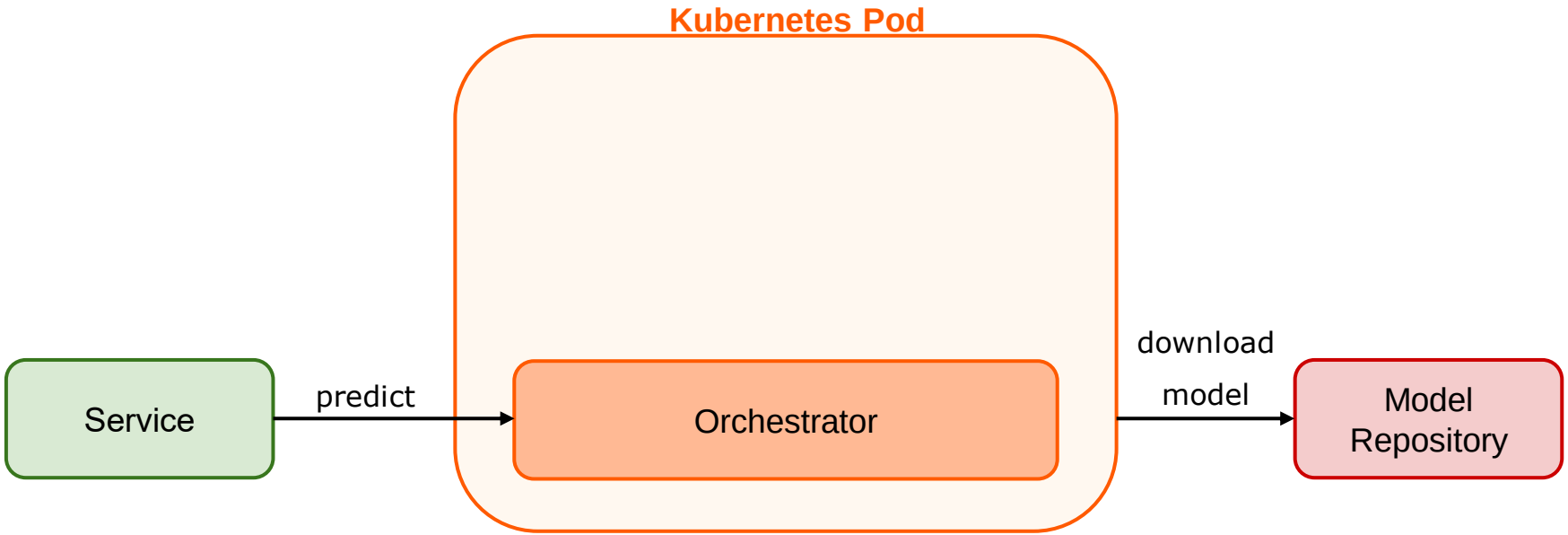


Model deployment



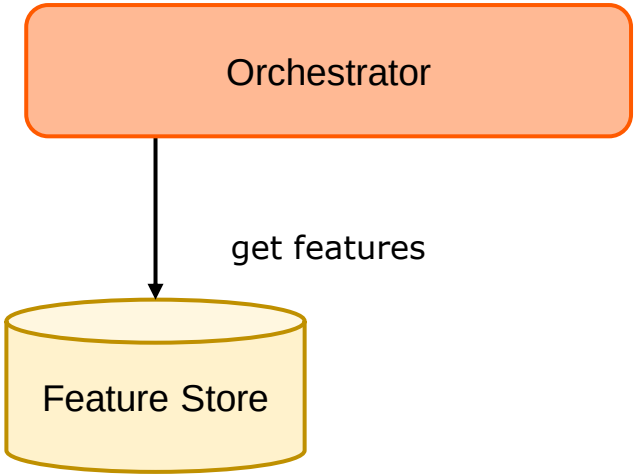

Model-service
Service
download
model
Model
Repository


Model-service
Service
predict
download
model
Model
Repository


Kubernetes Pod
Service
predict
Orchestrator
download
model
Model
Repository

Orchestrator

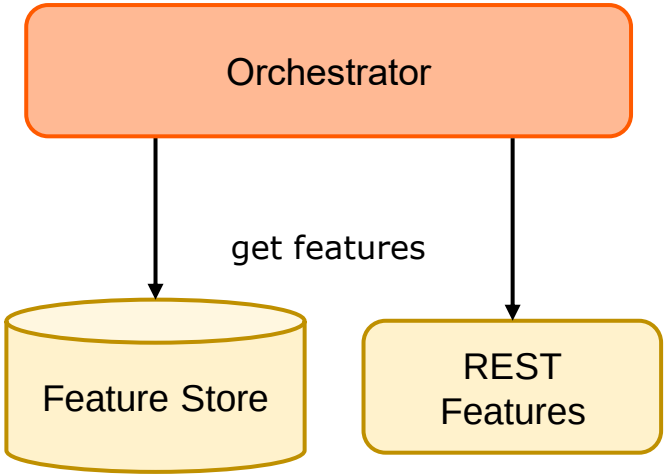
Requests features from Feature Store



Orchestrator

Requests features from Feature Store

Requests REST features

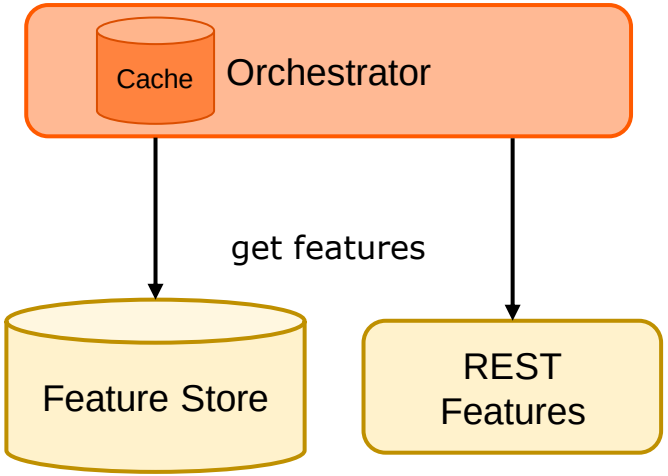


Orchestrator

Requests features from Feature Store

Requests REST features

Caches features



Orchestrator

Requests features from Feature Store

Requests REST features

Caches features

Declarative configuration

Orchestrator

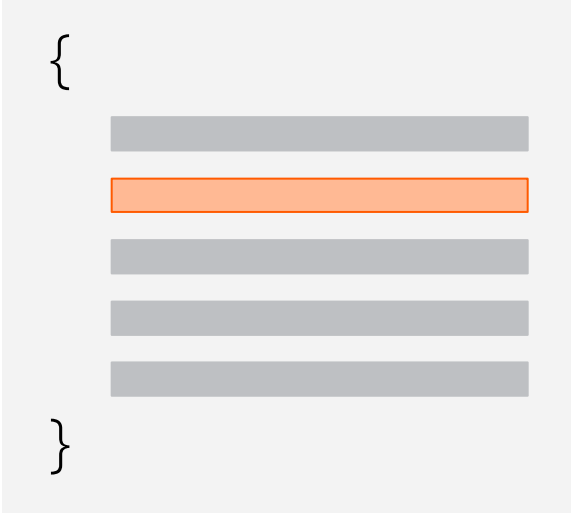
Requests features from Feature Store

Requests REST features

Caches features

Declarative configuration

- Features per object



Orchestrator

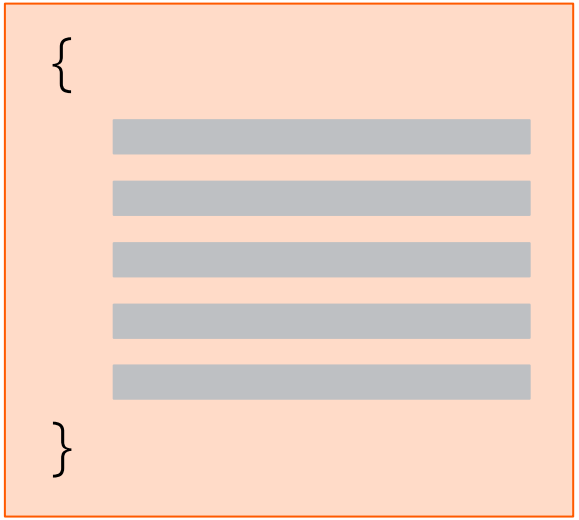
Requests features from Feature Store

Requests REST features

Caches features

Declarative configuration

- Features per object
- Features per request



Orchestrator

Requests features from Feature Store

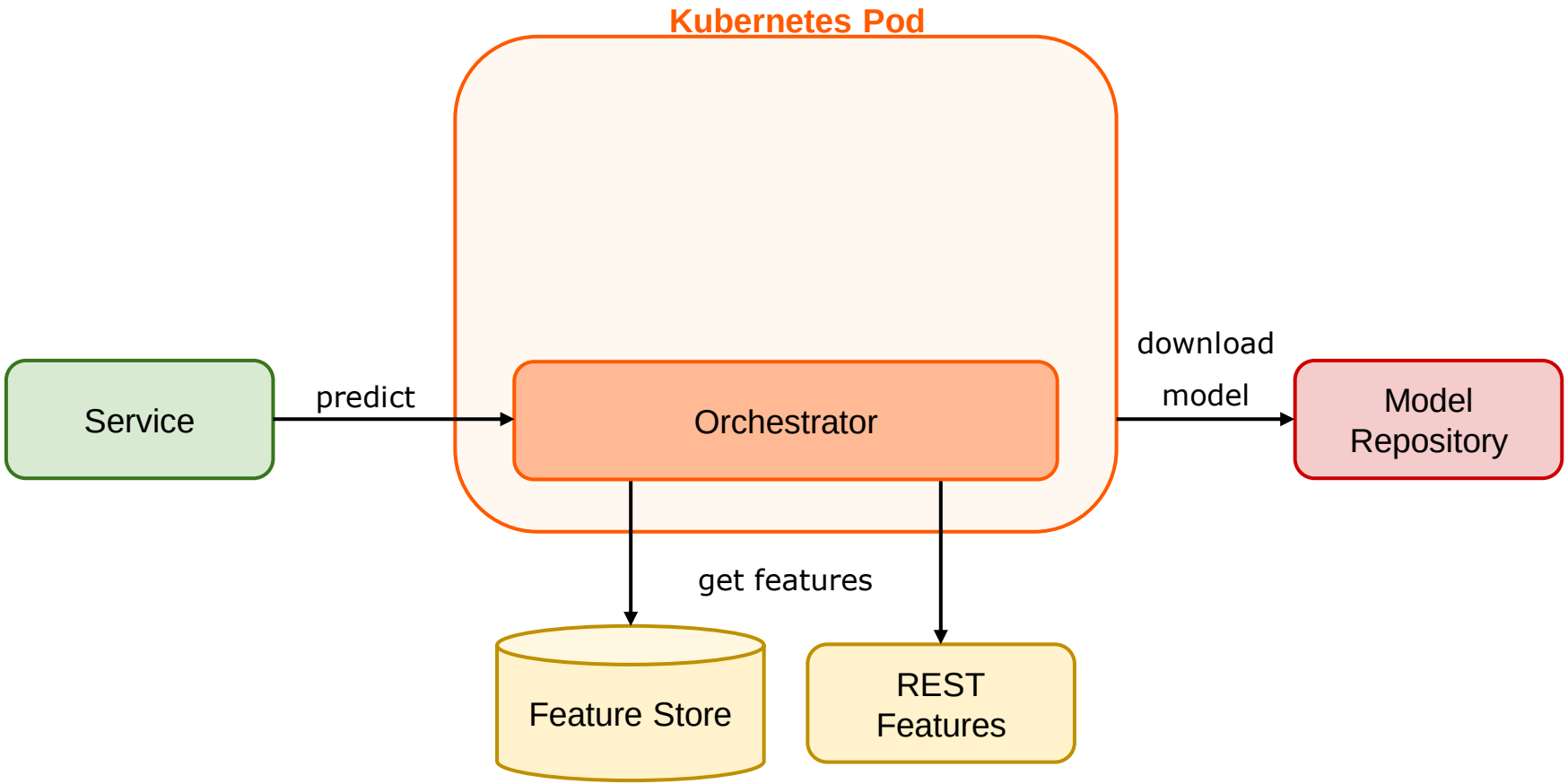
Requests REST features

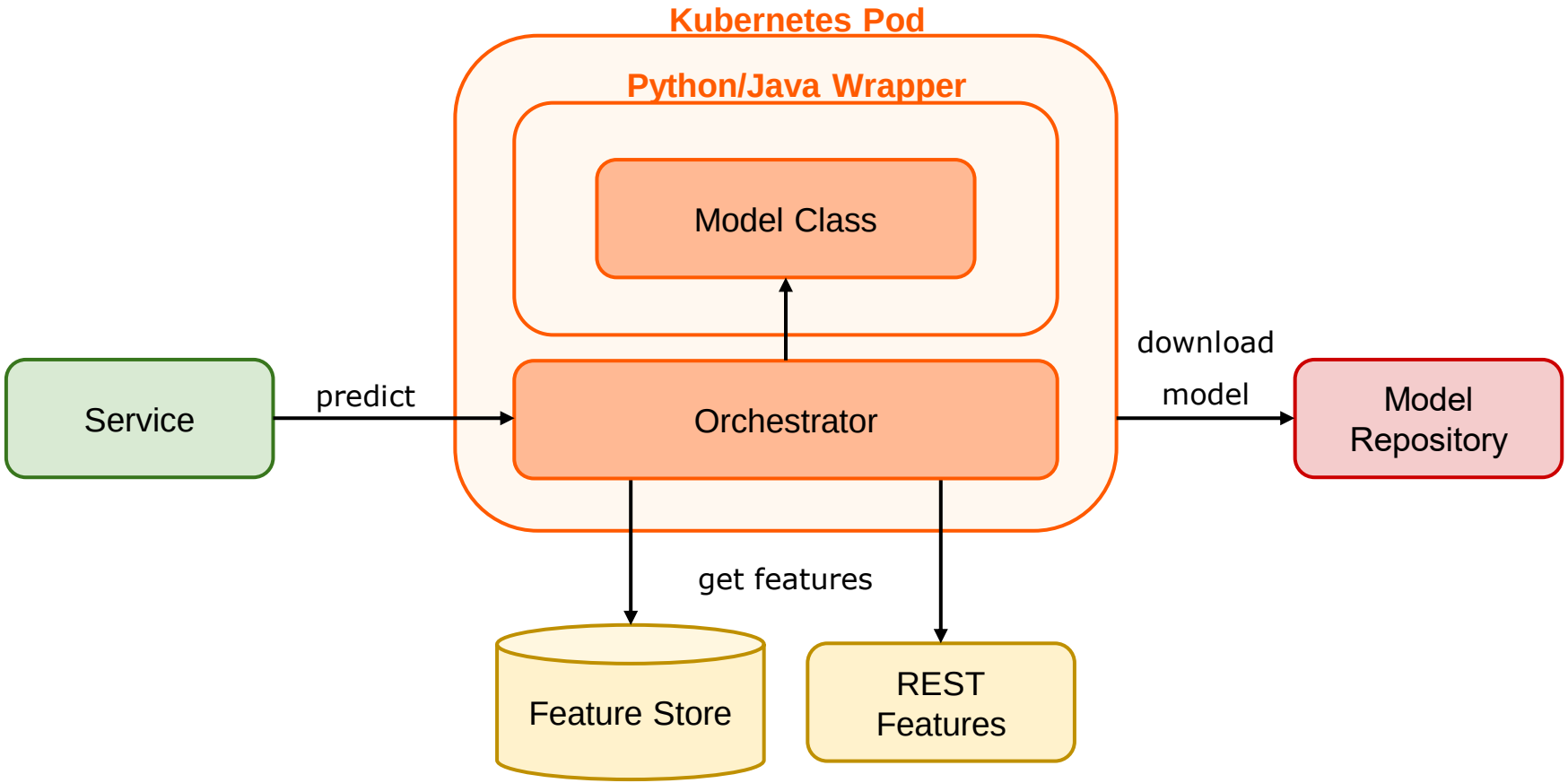
Caches features

Declarative configuration

- Features per object
- Features per request

Gathers metrics


Kubernetes Pod
Service
predict
Orchestrator
download
model
Model
Repository
get features
Feature Store
REST
Features


Kubernetes Pod
Python/Java Wrapper
Model Class
Service
predict
Orchestrator
download
model
Model
Repository
get features
Feature Store
REST
Features

Model Wrapper

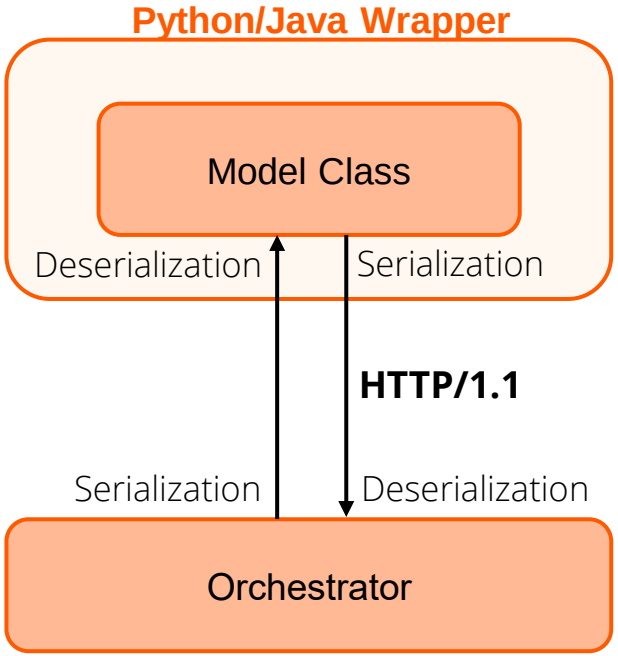
Supports

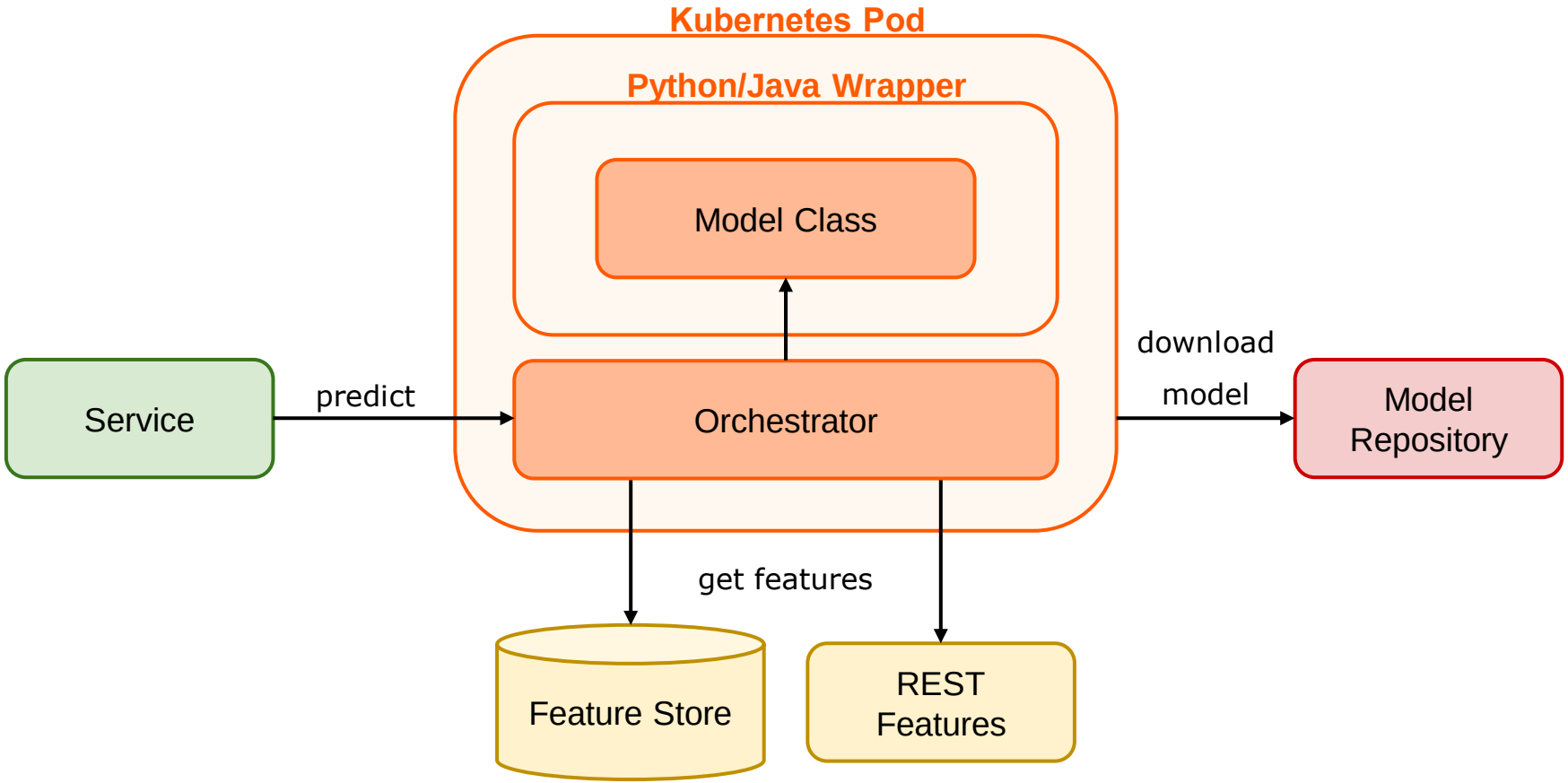
Interface

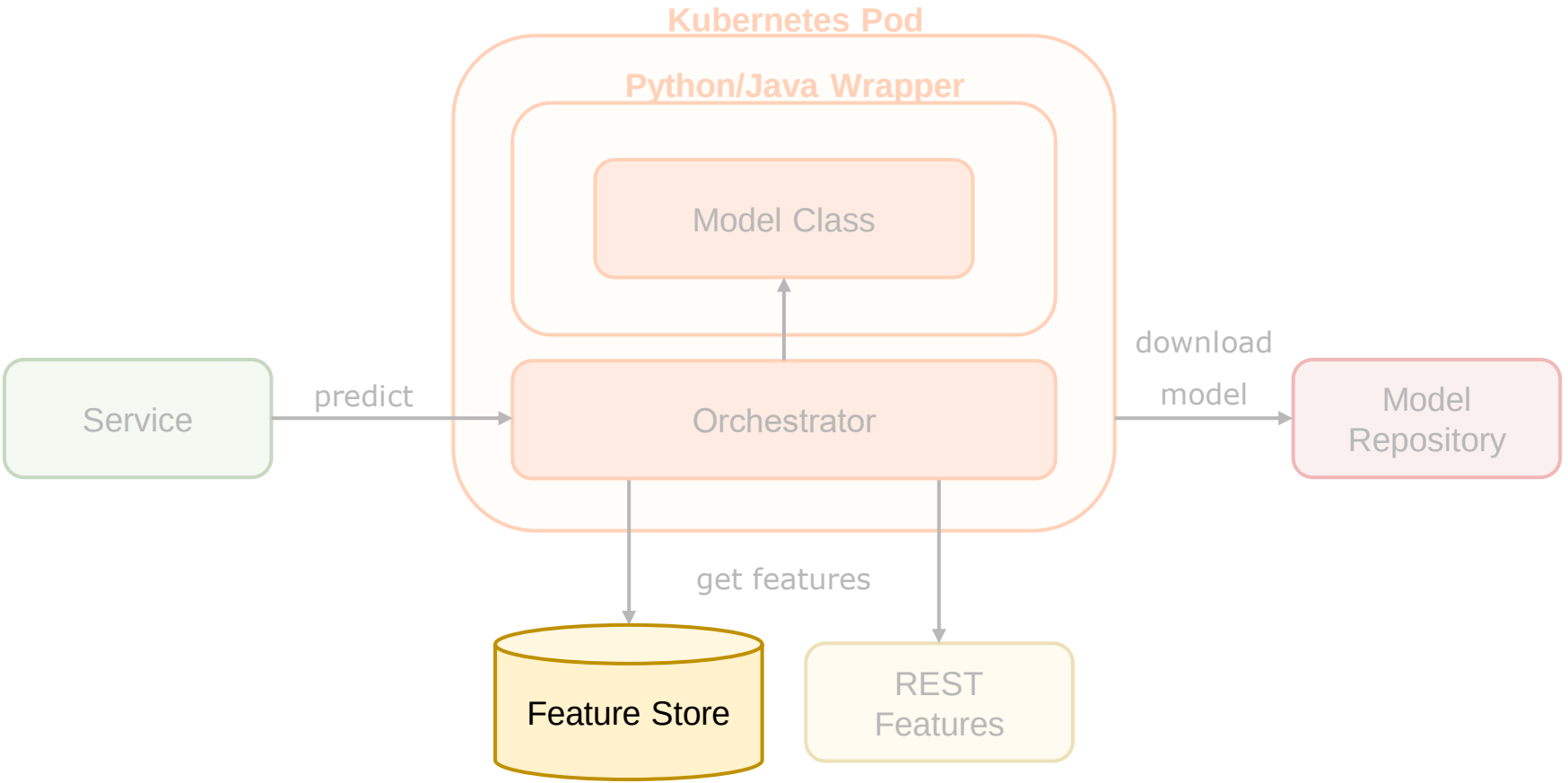
```
class MyPredictor:
    def predict(self, request):
                    pass

    def load(self, model_dir):
                    pass
```

Orchestrator/Model Wrapper Communication




Kubernetes Pod
Python/Java Wrapper
Model Class
Service
predict
Orchestrator
download
model
Model
Repository
get features
Feature Store
REST
Features


Kubernetes Pod
Python/Java Wrapper
Model Class
Service
predict
Orchestrator
download
model
Model
Repository
get features
Feature Store
REST
Features

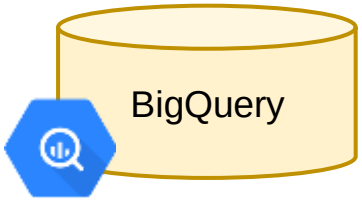
Feature Store Requirements

Low latency storage

Seamless integration with online prediction

Handling a lot of traffic

Easy to use by Researchers


BigQuery

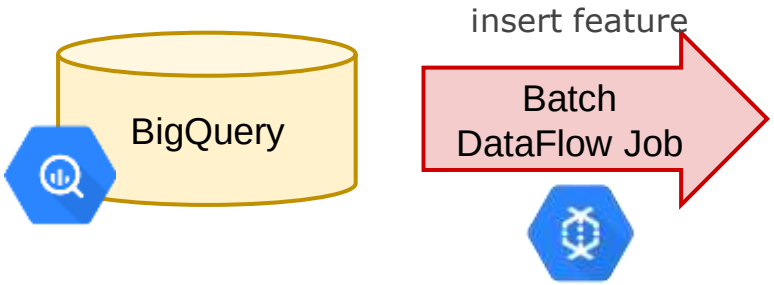
Feature Store Offline

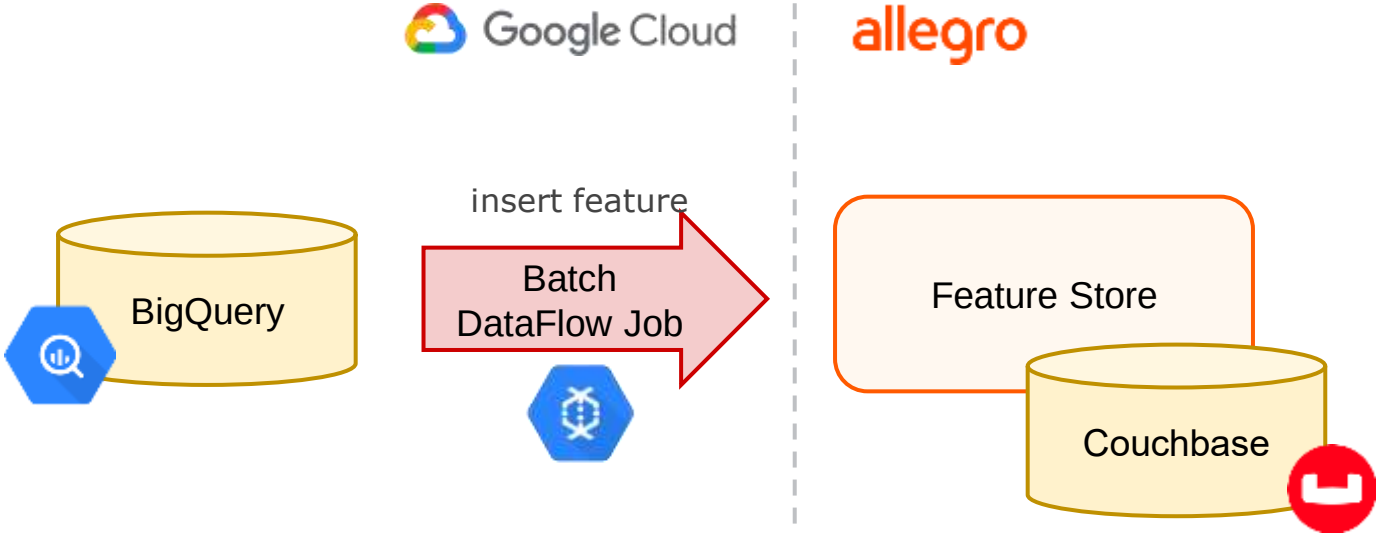
Why BigQuery:

- Commonly used in allegro
- Easy to use for Researchers
- Cheap and good for analytics

Ingestion pipeline:

- User responsibility
- Technology agnostic


BigQuery
insert feature
Batch
DataFlow Job


Google Cloud
allegro
BigQuery
insert feature
Batch
DataFlow Job
Feature Store
Couchbase

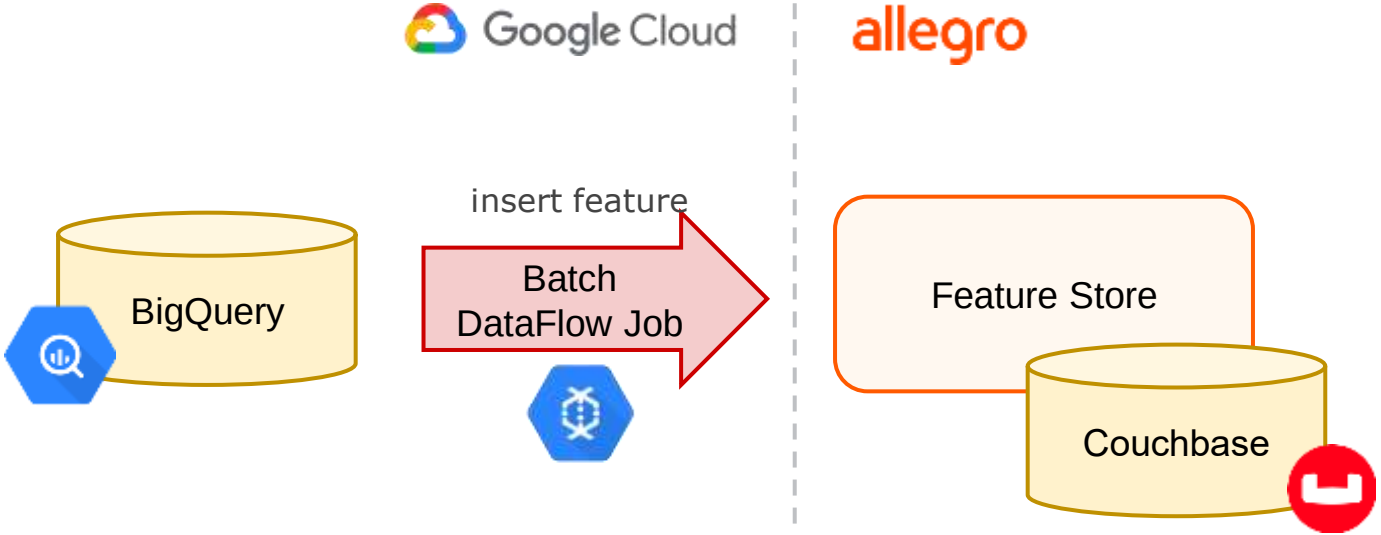
Couchbase

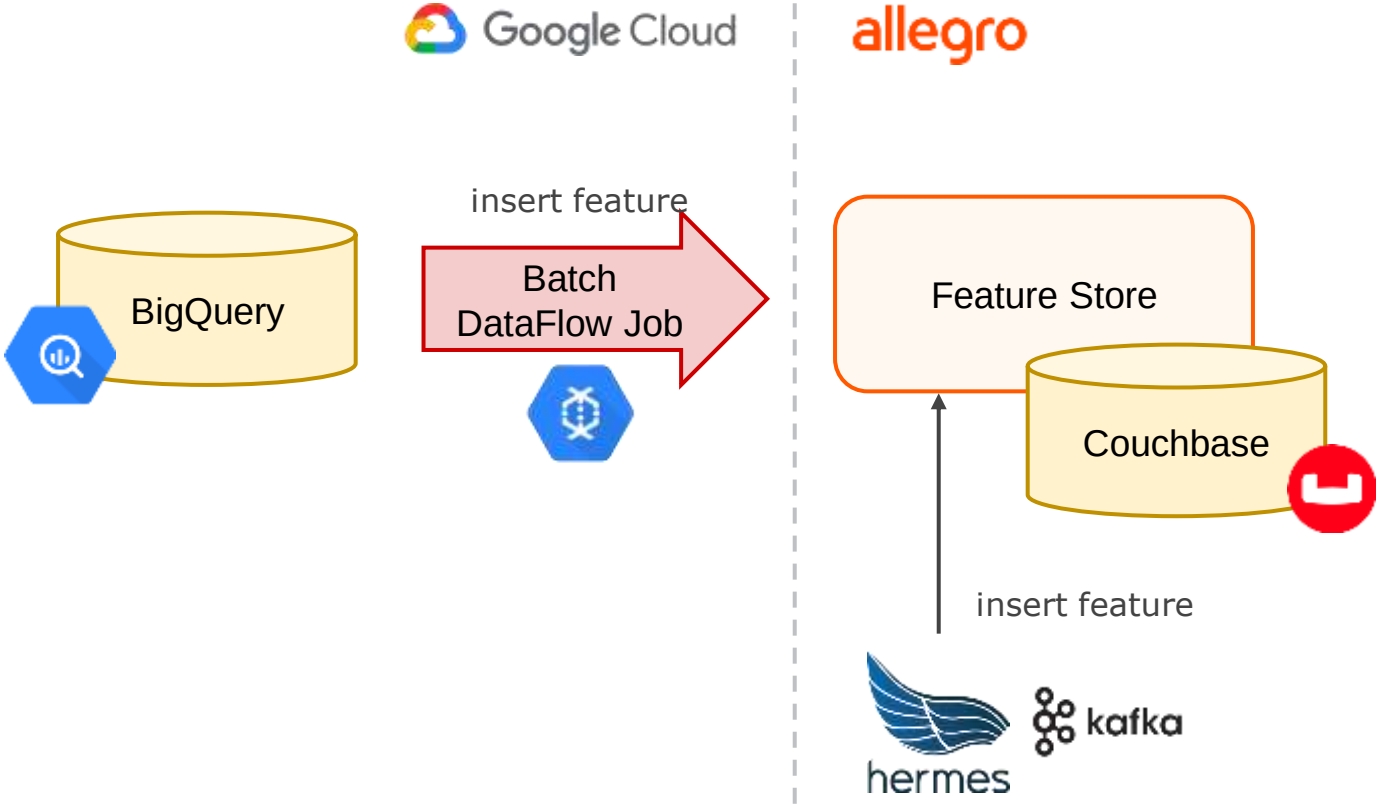
Low latency (< 10ms p95)

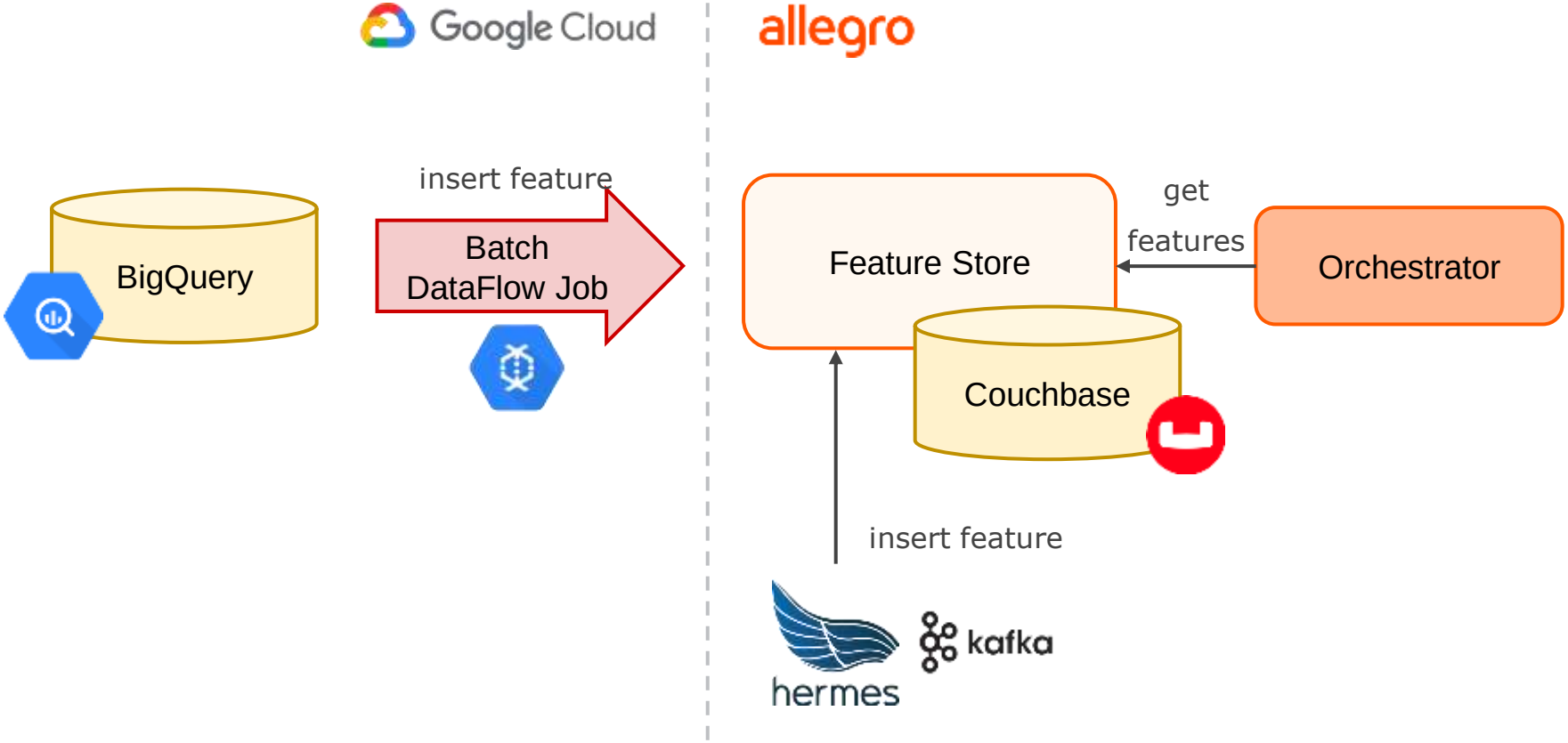
Handles huge traffic (hundreds of thousands reads per second)

Already used in our organization

Not perfect for scalability - requires manual scaling


Google Cloud
allegro
BigQuery
insert feature
Batch
DataFlow Job
Feature Store
Couchbase


Google Cloud
allegro
BigQuery
insert feature
Batch
DataFlow Job
Feature Store
Couchbase
insert feature
hermes
kafka


Google Cloud
allegro
BigQuery
insert feature
Batch
DataFlow Job
Feature Store
get
features
Orchestrator
Couchbase
insert feature
hermes
kafka

Lessons Learned

Sharing features

- Maintaining features for other teams
- Forces compatibility and limits experimentation
- Features meaning differ from team to team

Offline features for model training

- Each change breaks historical consistency of the feature
- Backfilling data on every change is really expensive
- Tables and queries for training differ from online

Custom vs Off-the-shelf

- Sometimes custom platform is easier to build than off-the-shelf one
- ... especially if you already have many tools available
- ... but Apps != ML so some new features need to be added

Summary

- Tens of models
- Thousands of predictions per second
- Hundreds of features in feature store
- Hundreds of thousands of reads from Feature Store

Thank you!