Serving ML Models at Scale with Kubeflow and Seldon-Core

- Based in Barclays Rise, London
- Participated in Barclays TechStars Accelerator

TensorFlow™ MEET UP LONDON

Product: Machine Learning Deployment on Kubernetes (https://github.com/SeldonIO/seldon-core)

ML Consultancy:

- ML applications FX/Equity Prediction
- Churn prediction

Overview

- Machine Learning on Kubernetes
- Machine Learning Deployment Challenges
  - Seldon-core
- Kubeflow integration
- End-to-End Machine Learning
  - Example

Goal: Help Data Science Project Teams Succeed

Data Scientist

- Analyzes the data
- Builds the predictive model
- Optimizes the model

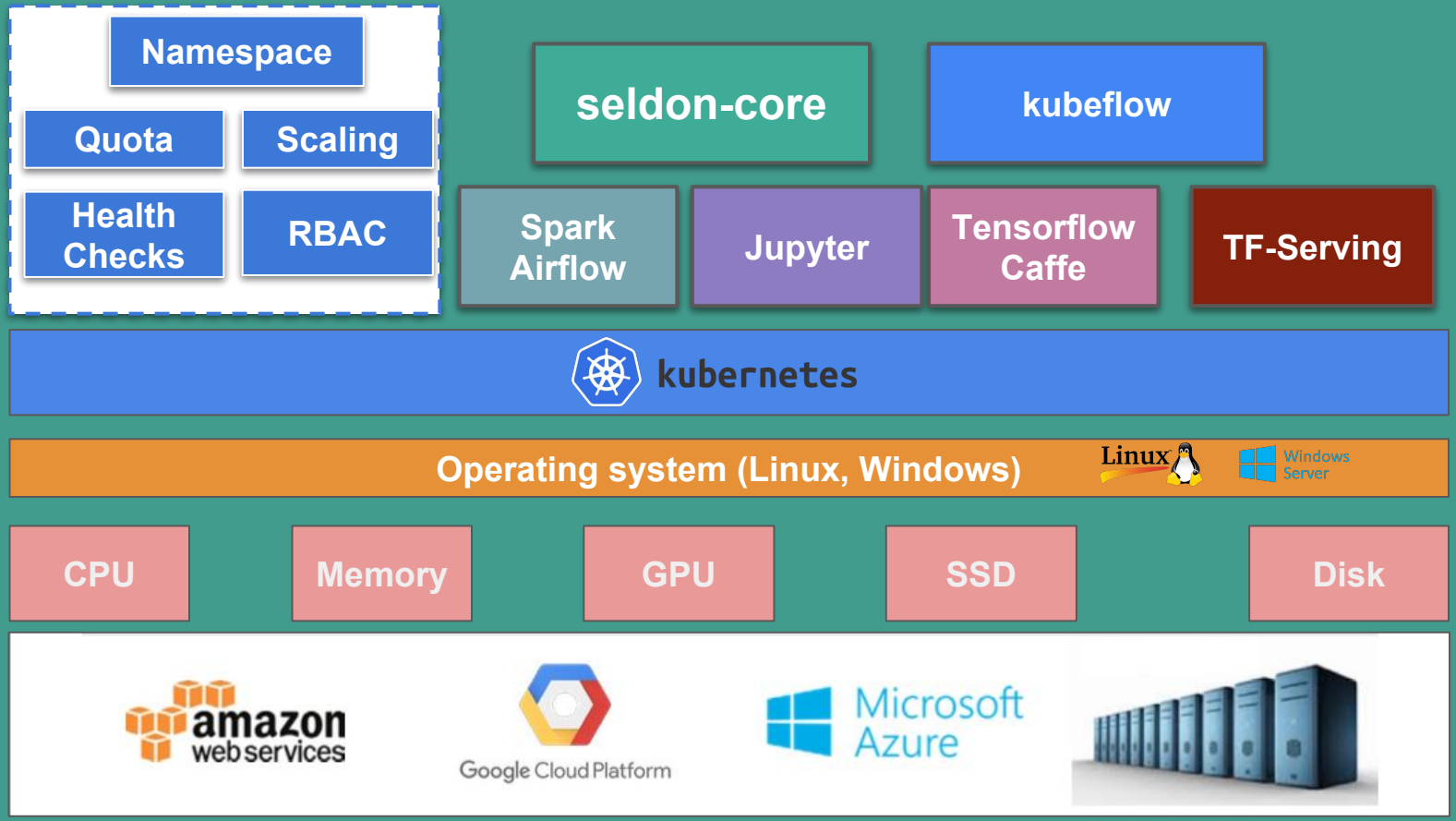
Data Engineer

- Manages infrastructure
- Monitors the model in production
- First response on issues

Business Manager

- Decides the project goals
- Defines business KPIs
- Evaluates ROI
- Provides Approval/Audits

Machine learning on Kubernetes



Machine Learning Deployment : Seldon Core

https://github.com/SeldonIO/seldon-core

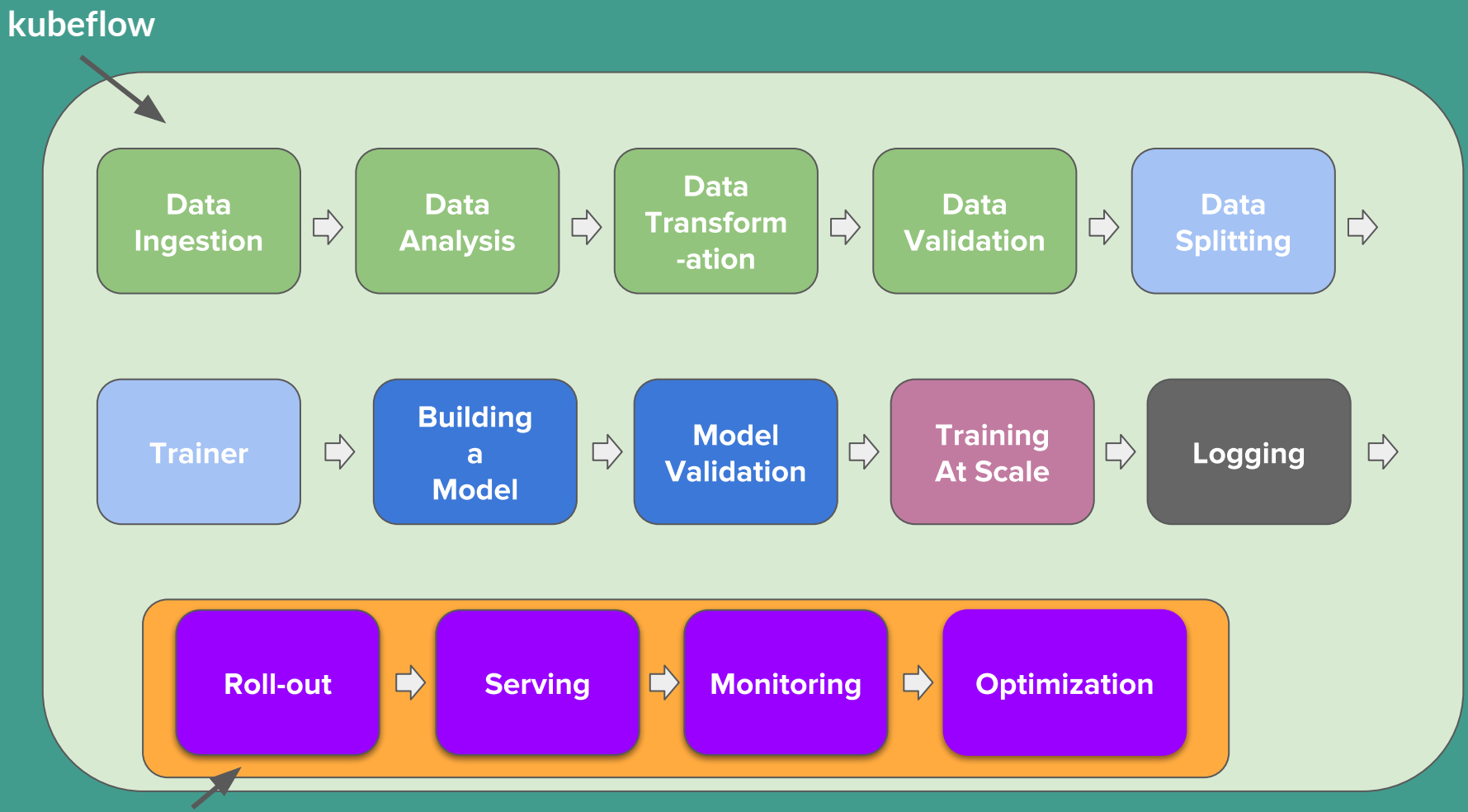
Seldon-Core Goals

- **Deployment**
  - Launch
  - Scaling up/down
  - Updates
    - Rolling
    - Canary
    - Blue-Green
    - Shadow
  - Health checks
  - Recovery
- **Optimization**
  - Infrastructure
  - Latency
  - Throughput
  - Model
- **Connect to Business Applications**
  - *Synchronous*
    - *REST*
    - *gRPC*
  - *Asynchronous*
    - *Message Queues*
- **Management**
  - Auditing
  - Versioning
  - Data provenance
  - Monitoring
  - CI/CD
    - “GitOps”

Seldon-Core Goals

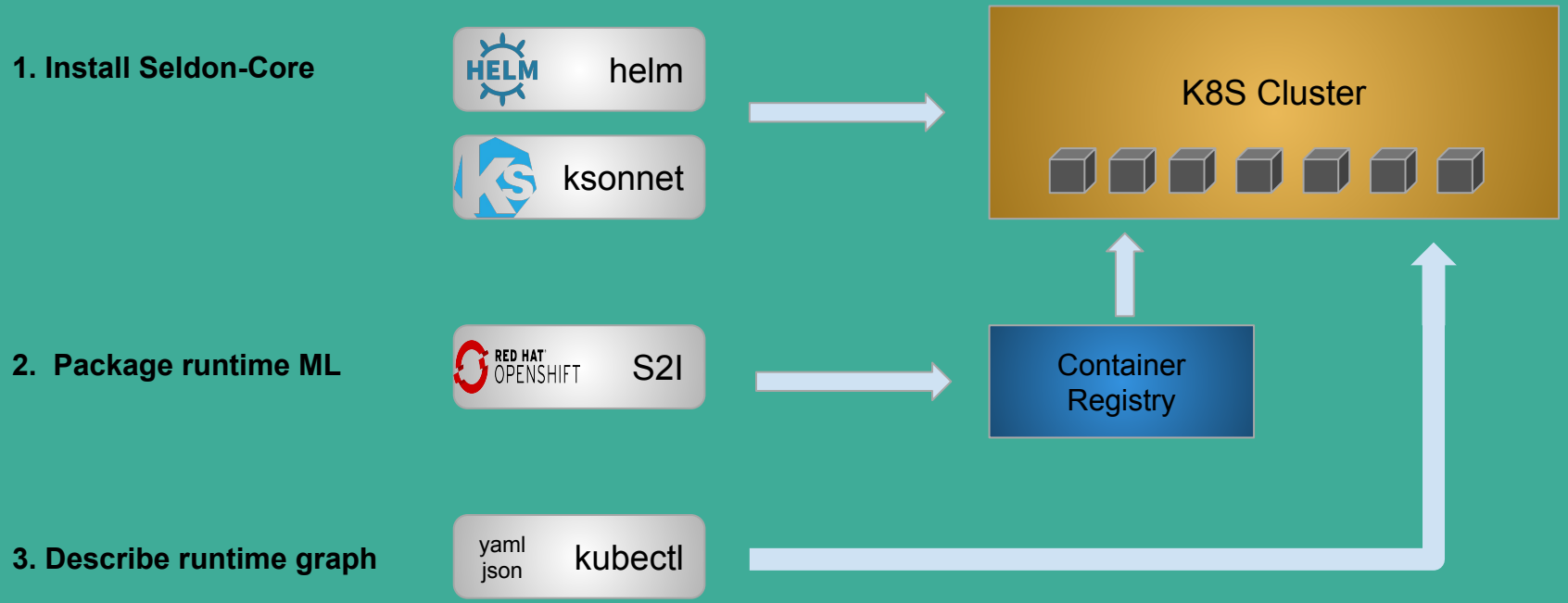
- **ML Tool Agnostic**
  - **Python**
    - TensorFlow
    - scikit-learn
  - **R**
  - **Java**
    - Spark
    - H2O
  - **Commercial Toolkits**

- **Dynamic ML Service Mesh**
  - *Routing requests*
    - *AB Tests*
    - *Multi-Armed Bandit*
  - *Transformations*
    - *Feature Normalization*
    - *Ensembles results*
  - Metrics
    - *Concept drift*
    - *Outlier detection*
  - *Security*

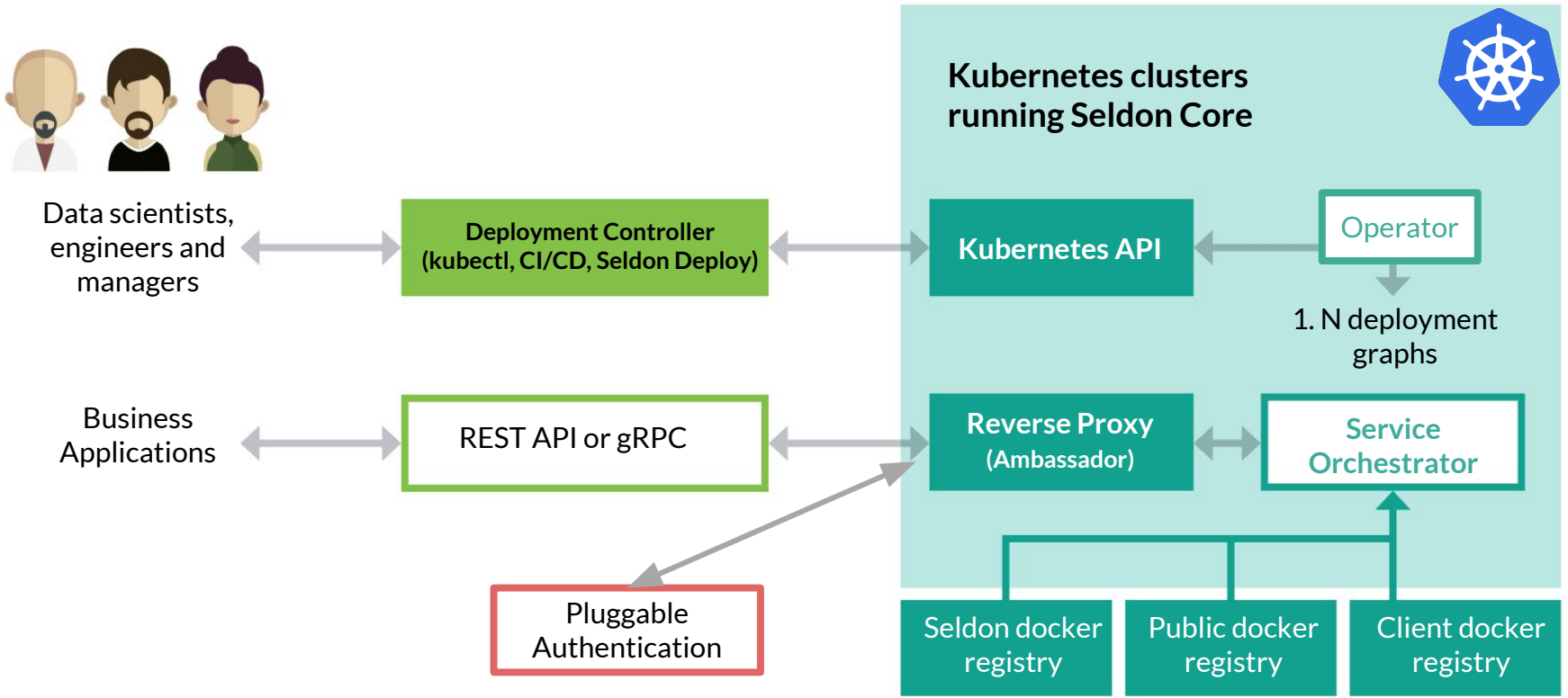


Courtesy kubeflow

Seldon-Core Machine Learning Deployment



seldon-core architecture



Runtime Prediction Graphs
Predictive Units

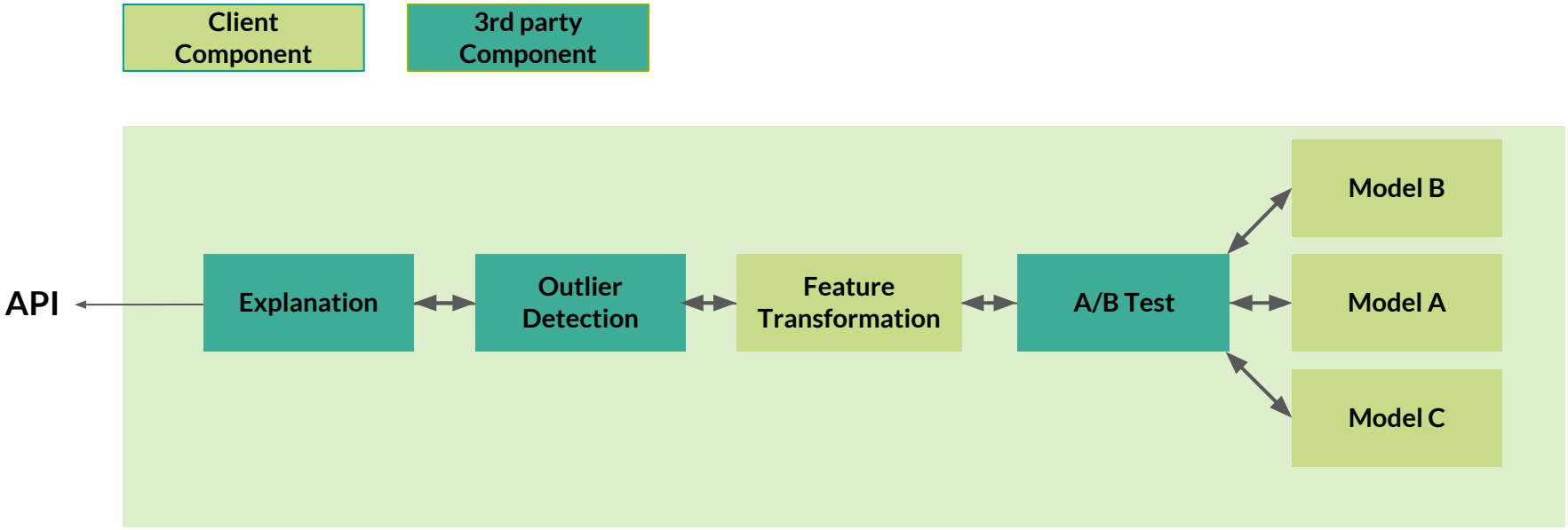
Models
- Runtime prediction models
  - *Tensorflow, sci-kit learn, H2O, Spark*

Routers
- Direct requests to one child graph
  - *A-B testing, Multi-Armed Bandits*

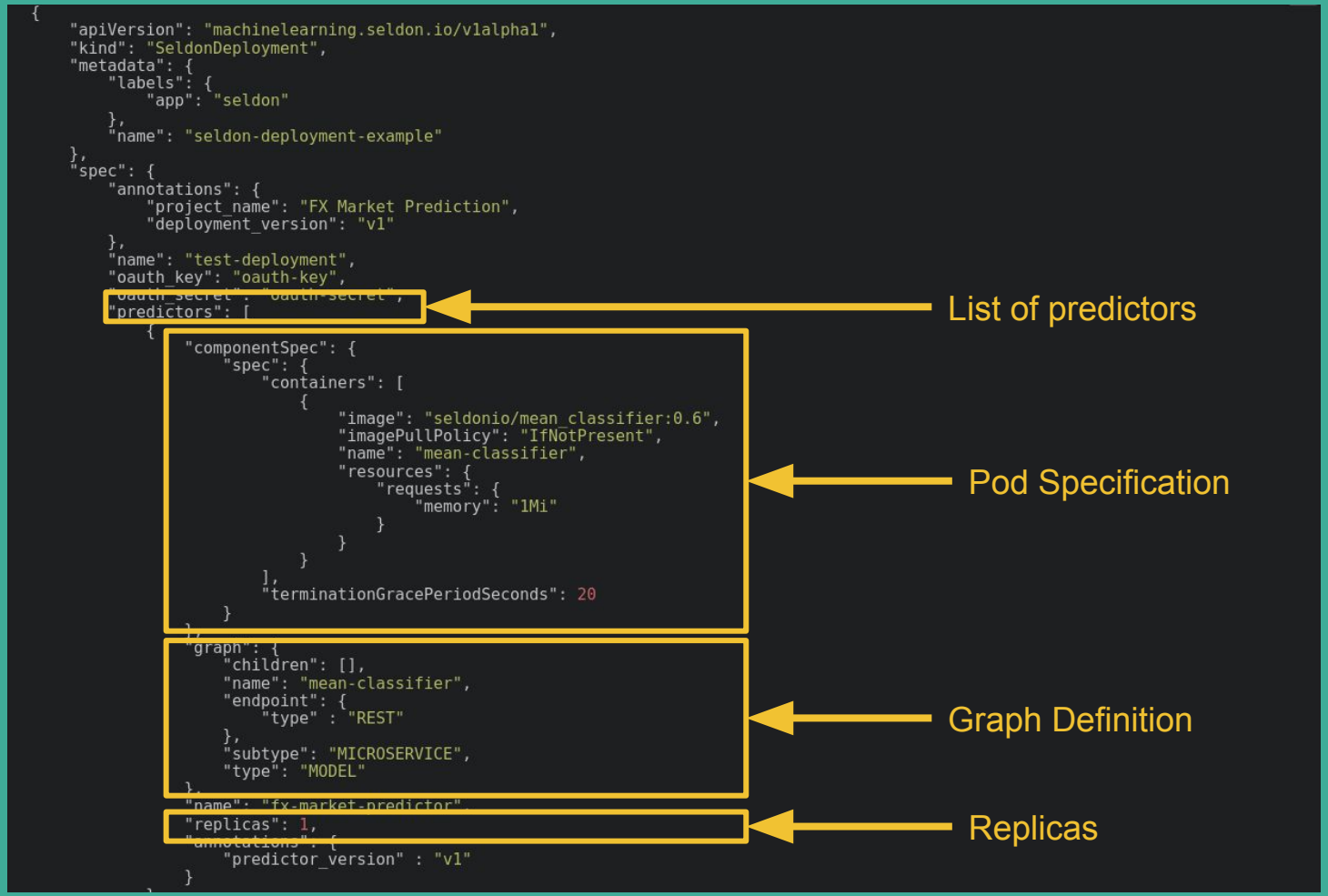
Combiners
- Combine responses from child graphs
  - *Ensemblers*

Transformers
- Transform the request
  - *Feature normalization*
- Transform response
  - *Concept drift, Outlier detection*

Seldon Core Complex Graphs

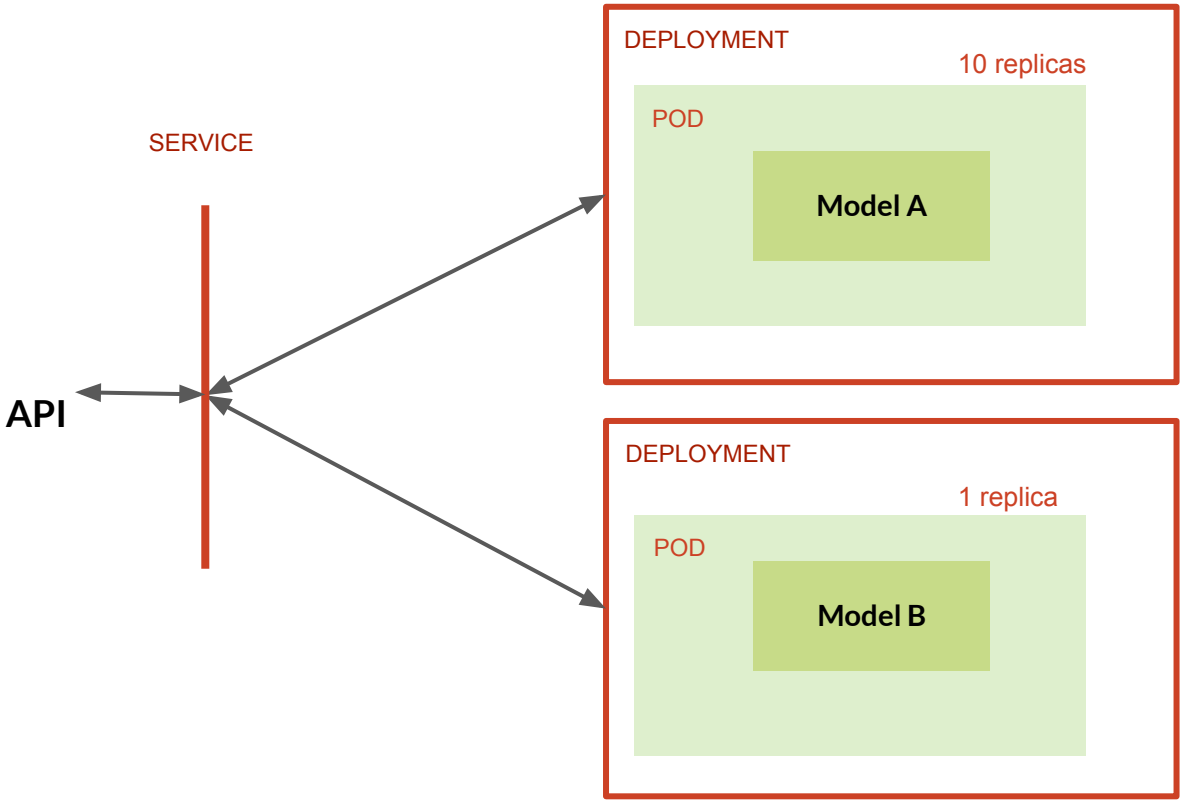


Example Seldon Deployment Manifest (custom kubernetes resource)

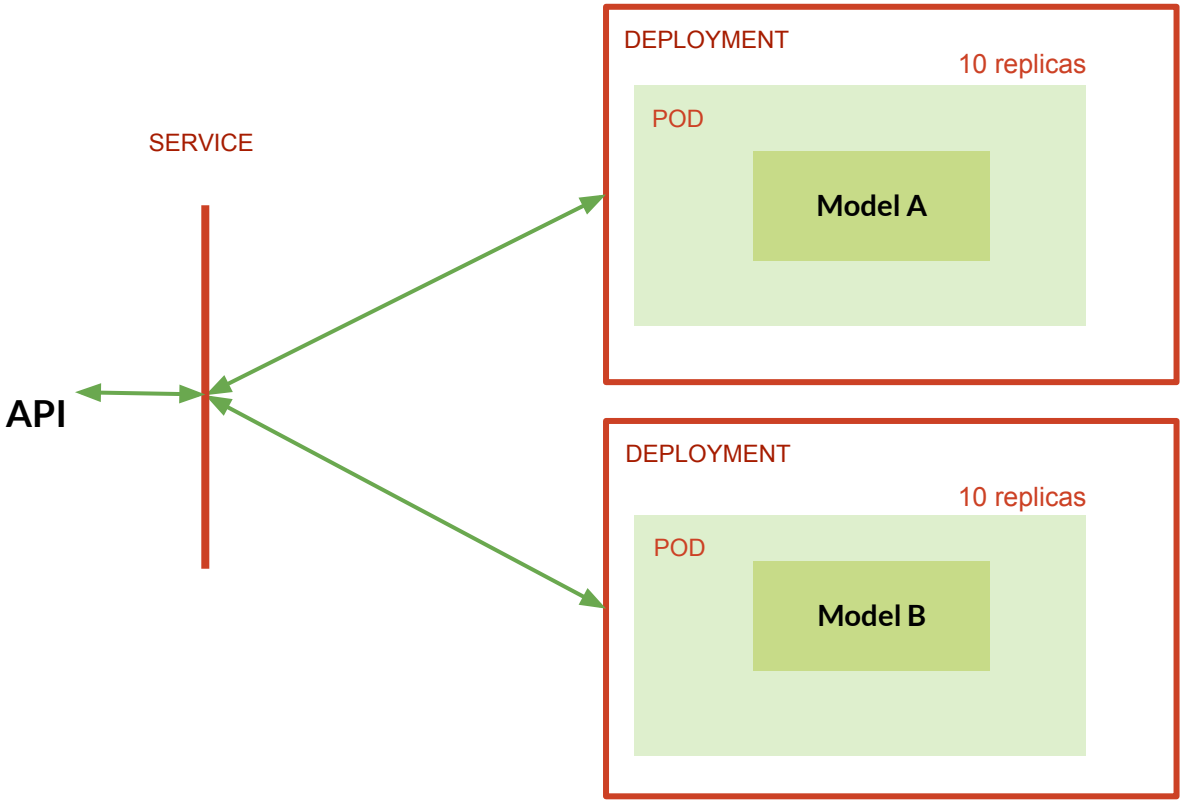


seldon

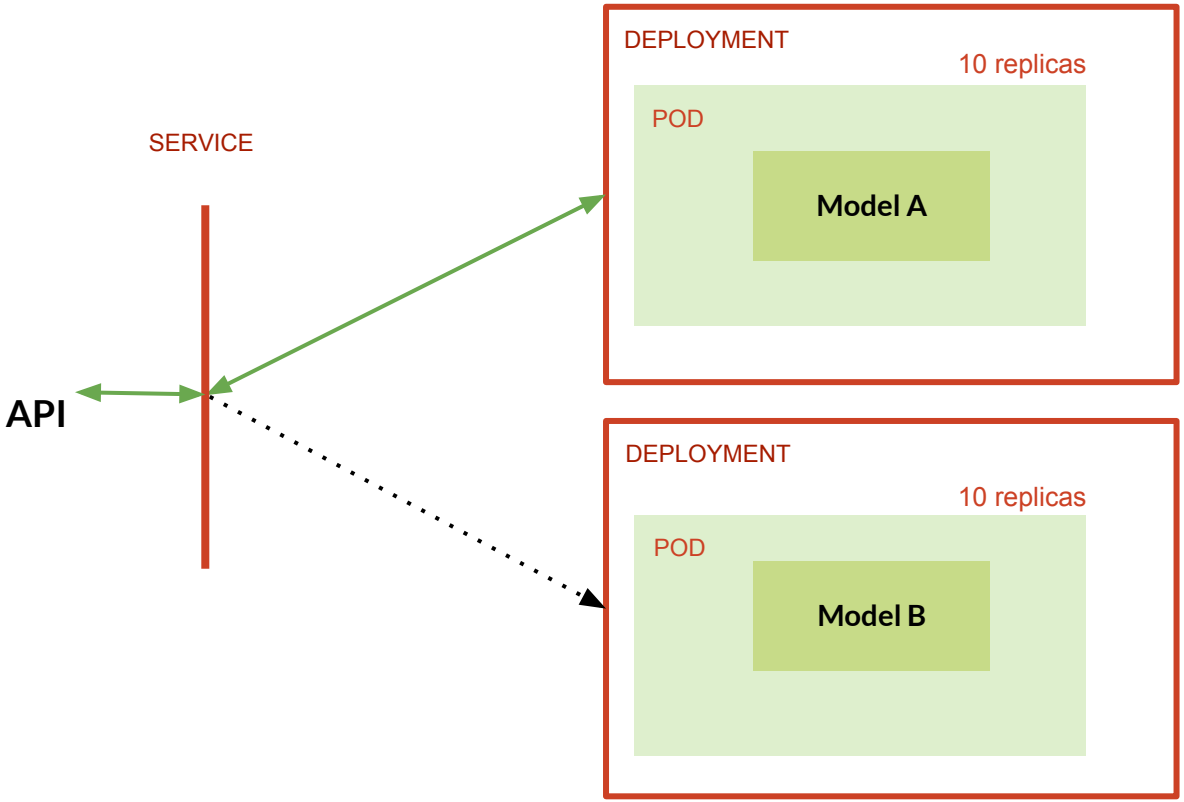
Canary


DEPLOYMENT
10 replicas
POD
Model A
SERVICE
API
DEPLOYMENT
1 replica
POD
Model B

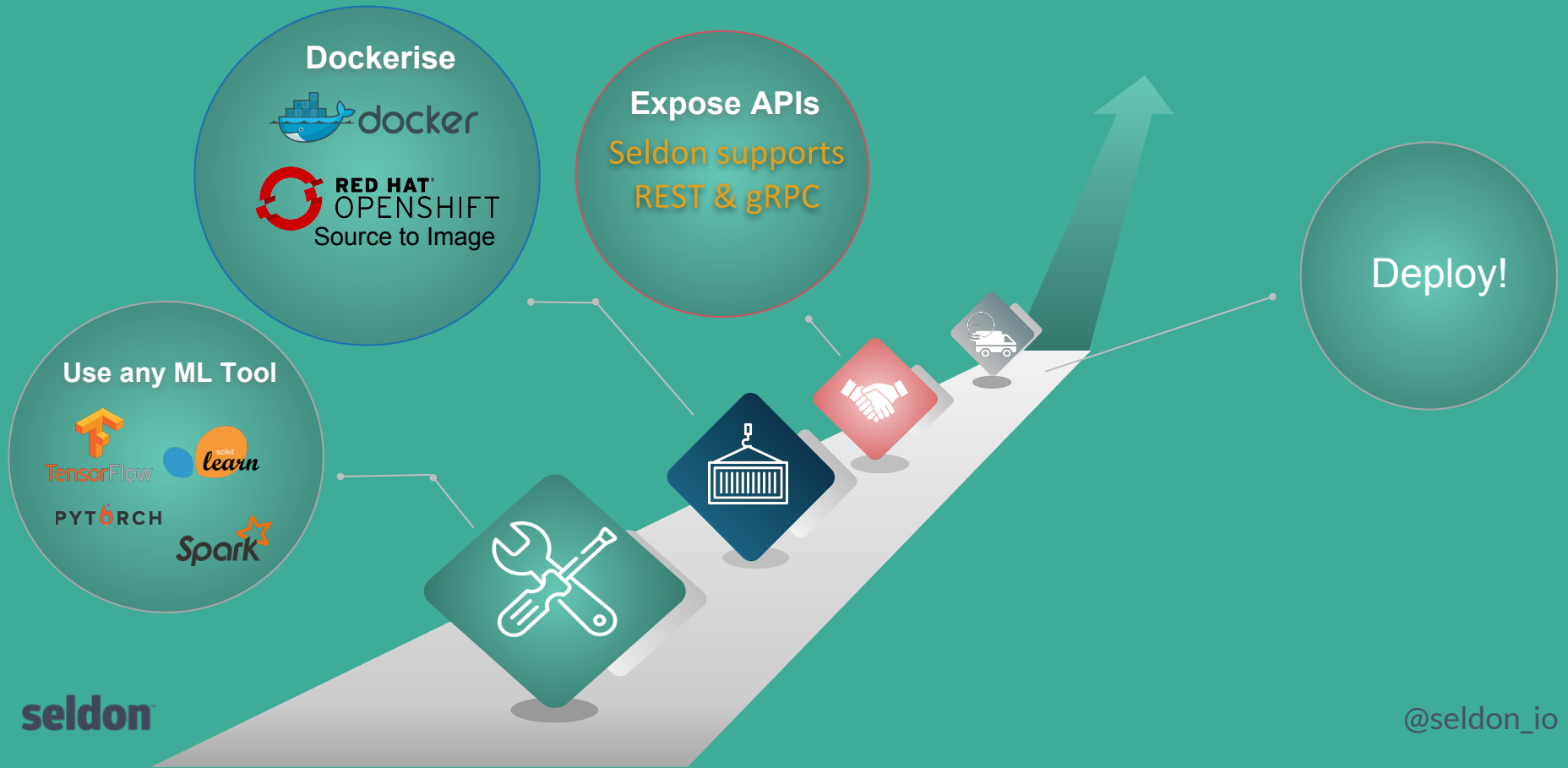
Blue-Green Deployments



Shadow Deployments



Seldon-Core ML Tool Agnostic



Openshift Source-to-Image

https://github.com/openshift/source-to-image

User Source Code (Github/Local)

```
from sklearn.externals import joblib

class IrisClassifier(object):

        def __init__(self):
            self.model =  joblib.load('/mnt/model/IrisClassifier.sav')

        def predict(self,X,features_names):
            return self.model.predict_proba(X)
```

+

Seldon Builder Images

seldonio/seldon-core-s2i-python2
seldonio/seldon-core-s2i-python3
seldonio/seldon-core-s2i-r
seldonio/seldon-core-s2i-java-build

- ./assemble
- ./run
- ./usage

Dependencies

S2I

User Image
Ready to deploy on seldon-core

Wrapping python models with S2I

Tensorflow, sklearn, pyTorch, etc.

RED HAT OPENSHIFT
Source to Image

IrisClassifier.py

```
from sklearn.externals import joblib

class IrisClassifier(object):

        def __init__(self):
            self.model = joblib.load('/mnt/model/IrisClassifier.sav')

        def predict(self,X,features_names):
            return self.model.predict_proba(X)
```

requirements.txt

```
scikit-learn==0.19.0
scipy==0.18.1
```

.s2i/environment

```
MODEL_NAME=IrisClassifier
API_TYPE=REST
SERVICE_TYPE=MODEL
```

```
s2i build . seldonio/seldon-core-s2i-python2 myrepo/iris-py-classifier
```

Wrapping R models with S2I

RED HAT® OPENSHIFT
Source to Image

iris.R

```
library(methods)

predict.iris <- function(iris,newdata=list()) {
  predict(iris$model, newdata = newdata)
}

new_iris <- function(filename) {
  model <- readRDS(filename)
  structure(list(model=model), class = "iris")
}

initialise_seldon <- function(params) {
  new_iris("model.Rds")
}
```

install.R

```
install.packages('rpart')
```

.s2i/environment

```
MODEL_NAME=iris.R
API_TYPE=REST
SERVICE_TYPE=MODEL
```

```
s2i build . seldonio/seldon-core-s2i-r myrepo/iris-r-classifier
```

Wrapping Java models with S2I

H2O, Spark (Enterprise), DL4J, Weka etc.

RED HAT® OPENSHIFT
Source to Image

pom.xml

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
<dependency>
   <groupId>io.seldon.wrapper</groupId>
   <artifactId>seldon-core-wrapper</artifactId>
   <version>0.0.1-SNAPSHOT</version>
</dependency>
```

MyModel.java

```
@Component
@Primary
public class H2OModelHandler implements SeldonModelHandler {
 @Override
 public SeldonMessage predict(SeldonMessage payload) {
 //Custom Predict method here
}
```

.s2i/environment

```
API_TYPE=REST
SERVICE_TYPE=MODEL
```

```
s2i build . seldonio/seldon-core-s2i-java-build myrepo/java-model --runtime-image
seldonio/seldon-core-s2i-java-runtime
```

Seldon Core Workflow

1. Package
Create REST or gRPC dockerized microservice .

2. Describe Deployment
Create/update kubernetes resource manifest for deployment graph.

3. Deploy
Manage and analyze the performance of live deployments.

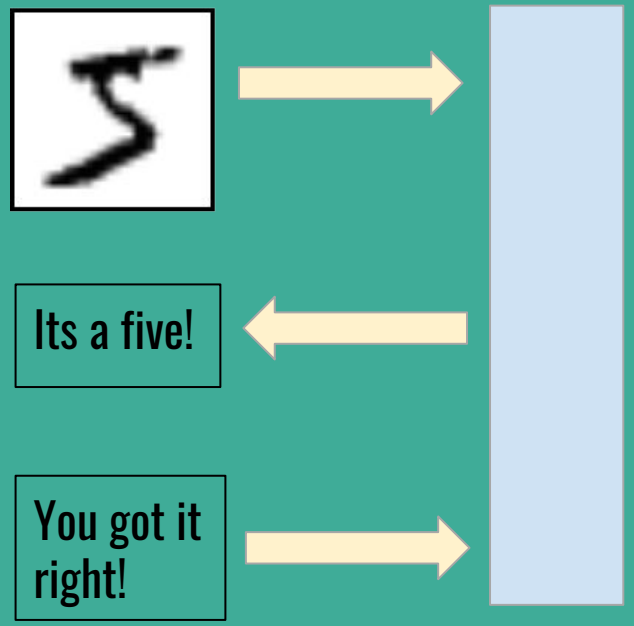
External API to connect to business
REST or gRPC

Predict

- Request/Responses generic payloads
- Data
  - *Tensor - shaped set of floats*
  - *NDArray - allow multi-typed and easy JSON serialization*
  - *Custom string or binary*
- Meta data

Feedback

- Request
- Response
- Reward



Seldon Core Roadmap

Low Latency

- Nvidia TensorRT
- Predictive batching
- Optimized single model scenarios

Data Provenance

- Add tags to wrapped models ; return in metadata
- Gitops

Distributed Graphs

- Multiple k8s deployments per graph
- Istio integration

Kubeflow

https://github.com/kubeflow/kubeflow

Kubeflow Components

Development

- Ksonnet Packages
- Jupyter Hub
- Tensorflow Training

Pipelines

- Argo Workflows

Deployment

- Ambassador reverse proxy
- Tensorflow Serving
- Seldon-core

Work In progress

- Batch Inference
- RPC Metrics
  - Tensorflow Serving
  - Seldon Core
- Integration ML Toolkits
  - MxNet
  - PyTorch
  - Pachyderm
- Central Dashboard

CRDs for TensorFlow, pyTorch, and more...

```yaml
apiVersion: "kubeflow.org/v1alpha1"
kind: "TFJob"
metadata:
  name: "example-job"
spec:
  replicaSpecs:
    - replicas: 1
      tfReplicaType: MASTER
      template:
        spec:
          containers:
            - image: gcr.io/tf-on-k8s-dogfood/tf_sample:dc944ff
              name: tensorflow
          restartPolicy: OnFailure
    - replicas: 1
      tfReplicaType: WORKER
      template:
        spec:
          containers:
            - image: gcr.io/tf-on-k8s-dogfood/tf_sample:dc944ff
              name: tensorflow
          restartPolicy: OnFailure
    - replicas: 2
      tfReplicaType: PS
      template:
        spec:
          containers:
            - image: gcr.io/tf-on-k8s-dogfood/tf_sample:dc944ff
              name: tensorflow
          restartPolicy: OnFailure
```

Using Kubeflow

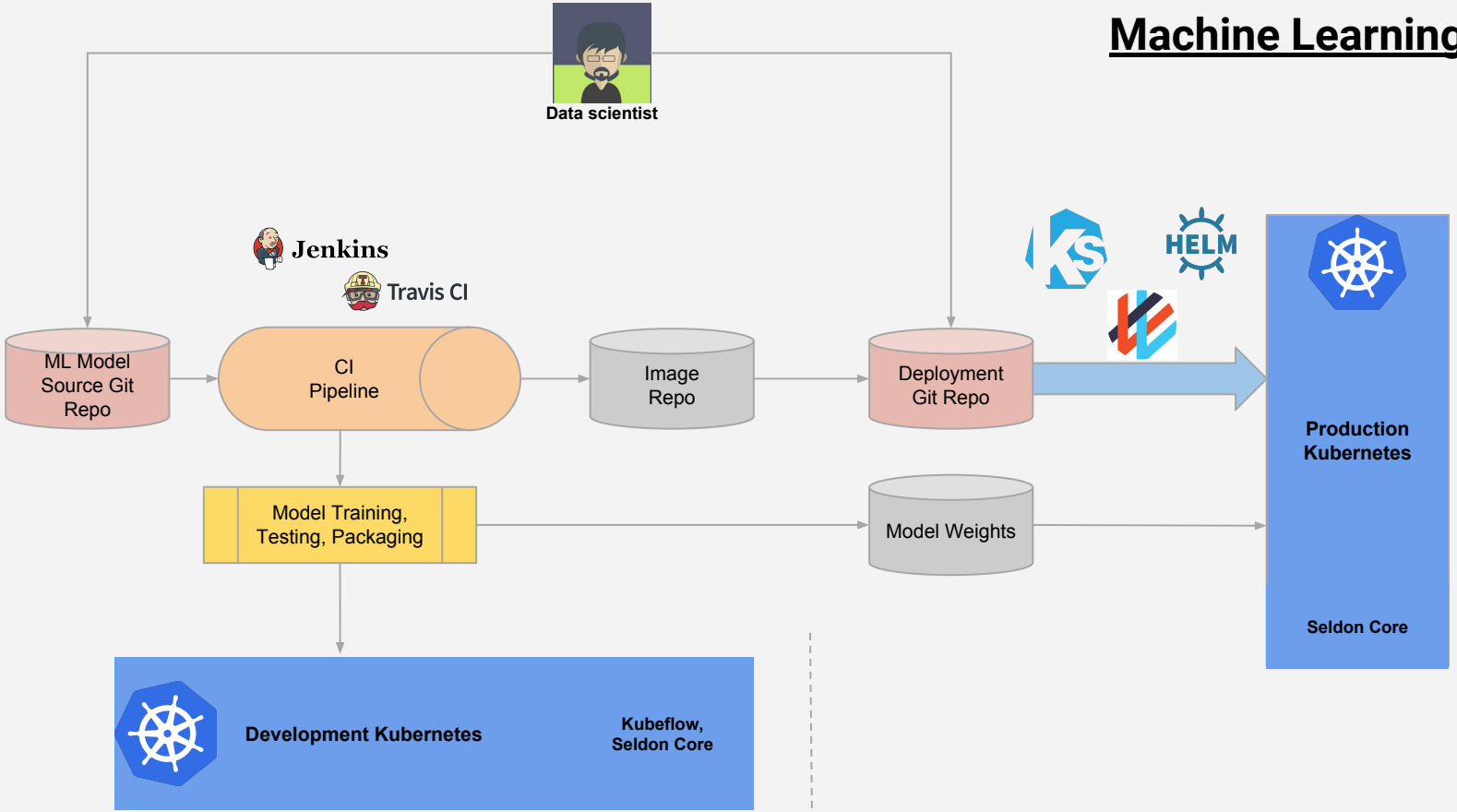
```
# Initialize a ksonnet APP
APP_NAME=my-kubeflow
ks init ${APP_NAME}
cd ${APP_NAME}

# Install Kubeflow components
ks registry add kubeflow github.com/kubeflow/kubeflow/tree/master/kubeflow
ks pkg install kubeflow/core
ks pkg install kubeflow/tf-job
ks pkg install kubeflow/tf-serving
ks pkg install kubeflow/seldon

# Deploy Kubeflow
NAMESPACE=kubeflow
kubectl create namespace ${NAMESPACE}
ks generate core kubeflow-core --name=kubeflow-core --namespace=${NAMESPACE}
ks apply default -c kubeflow-core
```

End-to-End Machine Learning

Machine Learning CI/CD



Continuous Integration

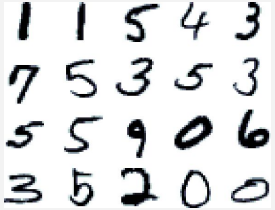
Continuous Deployment

Inspired by : https://www.weave.works/blog/the-gitops-pipeline

End-to-End ML Example

https://github.com/kubeflow/example-seldon

MNIST



Goal:
Classify Digits

Create Training Model
Images and Train

TensorFlow

scikit learn

argo

R

NFS

Model parameters

Create Runtime Model
Images

TensorFlow

scikit learn

argo

R

RED HAT OPENSHIFT
Source to Image

Deploy AB Test

Deploy single model

Deploy Multi-Armed bandit

seldon

@seldon_io

Inspired by : https://www.weave.works/blog/the-gitops-pipeline

Thank You

https://github.com/SeldonIO/seldon-core

https://github.com/kubeflow/example-seldon

https://github.com/kubeflow/kubeflow

Clive Cox
CTO

cc@seldon.io

Seldon Technologies Ltd

hello@seldon.io
seldon.io

US: +1 (646) 397-9911
UK: +44 (20) 7193-6752

41 Luke Street
London EC2A 4DP