Serving ML Models at Scale with Kubeflow and Seldon-Core
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

- Namespace
- Quota
- Scaling
- Health Checks
- RBAC

- seldon-core
- kubeflow
- Spark Airflow
- Jupyter
- Tensorflow Caffe
- TF-Serving

- kubernetes

- Operating system (Linux, Windows)

- CPU
- Memory
- GPU
- SSD
- Disk

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure
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**
Seldon-Core Machine Learning Deployment

1. Install Seldon-Core
   - helm
   - ksonnet

2. Package runtime ML
   - S2I

3. Describe runtime graph
   - yaml
   - json

K8S Cluster

Container Registry
seldon-core architecture

Data scientists, engineers and managers

Business Applications

Deployment Controller (kubectl, CI/CD, Seldon Deploy)

REST API or gRPC

Pluggable Authentication

Kubernetes clusters running Seldon Core

Kubernetes API

Operator

1. N deployment graphs

Reverse Proxy (Ambassador)

Service Orchestrator

Seldon docker registry

Public docker registry

Client docker registry

seldon.io
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

- **Client Component**
- **3rd party Component**

**API**

- **Explanation**
- **Outlier Detection**
- **Feature Transformation**
- **A/B Test**

**Models**
- Model A
- Model B
- Model C
Example Seldon Deployment Manifest (custom kubernetes resource)
Canary

- **API**
- **SERVICE**
- **DEPLOYMENT**
  - **POD**
    - Model A (10 replicas)
  - **POD**
    - Model B (1 replica)
Blue-Green Deployments

API

SERVICE

DEPLOYMENT
10 replicas

POD
Model A

DEPLOYMENT
10 replicas

POD
Model B
Shadow Deployments

- Model A
  - POD
  - 10 replicas
- Model B
  - POD
  - 10 replicas

API ➔ SERVICE ➔ DEPLOYMENT ➔ POD ➔ Model A/Model B ➔ API
Seldon-Core ML Tool Agnostic

- Use any ML Tool
  - TensorFlow
  - PyTorch
  - Spark

- Dockerise

- Expose APIs
  - Seldon supports REST & gRPC

- Source to Image

- Deploy!
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)
Wrapping python models with S2I
Tensorflow, sklearn, pyTorch, etc.

IrisClassifier.py

```python
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

**iris.R**

```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**

```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.

```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>
</dependencies>
```

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

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

API_TYPE=REST
SERVICE_TYPE=MODEL
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

Its a five!
You got it right!
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 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

- Data scientist
- Jenkins
- Travis CI
- ML Model Source Git Repo
- CI Pipeline
- Image Repo
- Deployment Git Repo
- Model Weights
- Production Kubernetes
- Seldon Core
- Development Kubernetes
- Kubeflow, Seldon Core

Model Training, Testing, Packaging

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

End-to-End ML Example
https://github.com/kubeflow/example-seldon

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


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