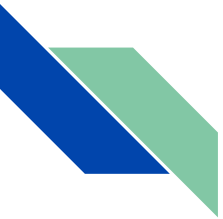


Kubeflow book!

http://introductiontomlwithkubeflow.com/

This is an intro level talk

Introducing Kubeflow

(w. Special Guests Tensorflow and Apache Spark)

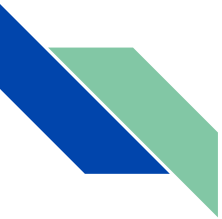
Agenda

- This is designed to be a **beginner** level talk
- About Me
- Background
- What Is KubeFlow?
- Kubeflow's Design and Core Components
- How to build cross-cloud ML with Kubeflow & Apache Spark
- How Spark & Tensorflow work together

Trevor Grant

- From Chicago
- Preferred pronouns he/him
- Various odd jobs around IBM (data scientist/evangelist/janitor)
- PMC Apache Mahout, Streams, Community Development
- IoT Track at ApacheCon North America
- Blog: rawkintrevo.org
- Twitter: @rawkintrevo
- Github: rawkintrevo
- Shameless self promotion: rawkintrevo.org/shameless-self-promotion/

[2] https://www.apachecon.com/acna19/index.html



Kubeflow Salesman:

Kubeflow Salesman:
Slaps roof of Kubeflow

Kubeflow Salesman:
Slaps roof of Kubeflow

Background

Things we thought you might need to know

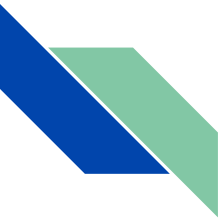
History of Predictive Analytics

Photo: Hans Splinter

Photo: Numerology Sign

Photo: Akash Kataruka

What is Statistics?



What is ~~Statistics?~~
Machine Learning?

What is ~~Statistics?~~

~~Machine Learning?~~

A.I. (Artificial Intelligence)

What is ~~Statistics?~~ ~~Machine Learning?~~ A.I. (Artificial Intelligence)

Model Training

Photo: Andreas Kretschmer

Model Serving

Photo: Helen Harrop

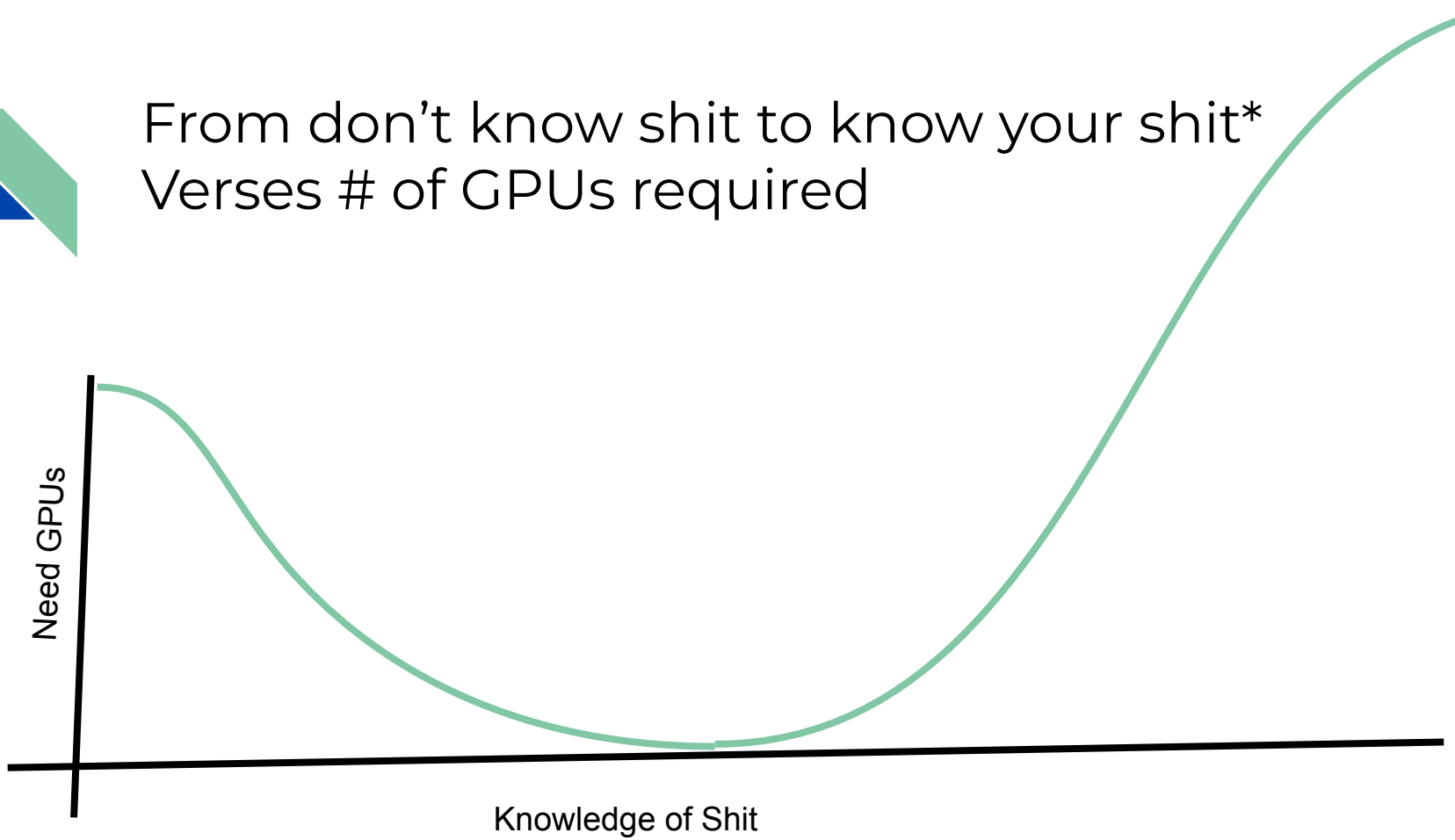
What is ~~Statistics?~~

~~Machine Learning?~~

A.I. (Artificial Intelligence)

+

From don’t know shit to know your shit*
Verses # of GPUs required



*Holden’s gut feelings after a reasonable amount coffee

What is Kubernetes?

+

Kubeflow: Dev on Laptop Deploy in Cloud

Deploy to Production

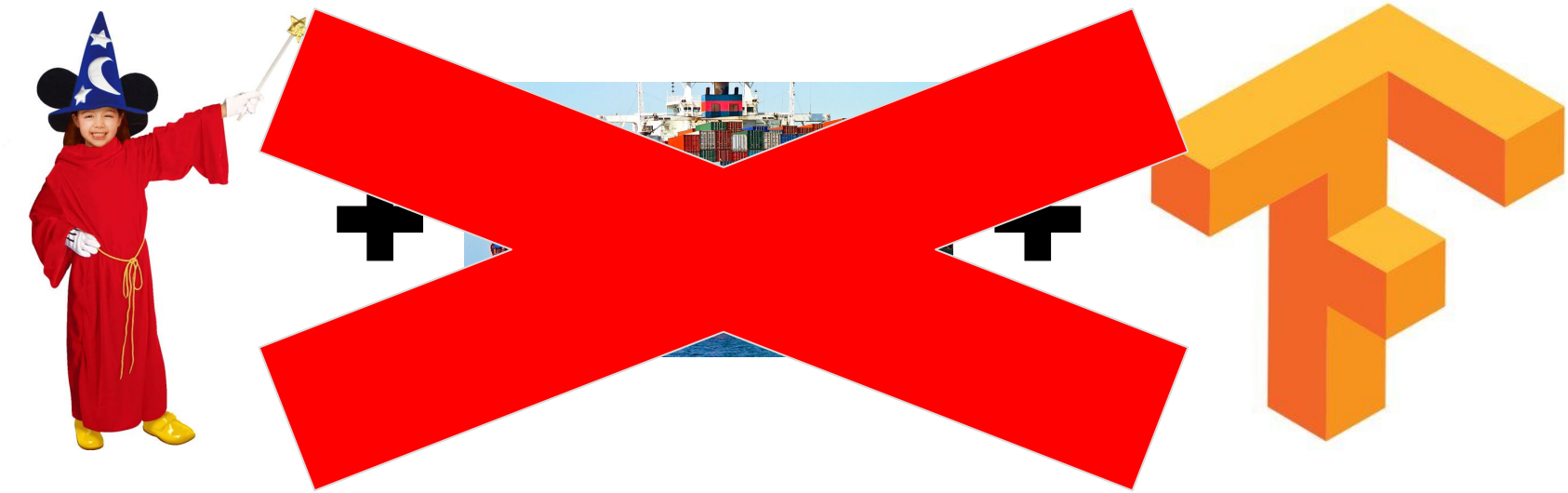
So what is Kubeflow?

What is Kubeflow?

+

+

What is Kubeflow?



What is Kubeflow?

Components Buffet

argo

automation

chainer-job

core

credentials-pod-preset

katib

mpi-job

mxnet-job

openmpi

pachyderm

pytorch-job

Seldon

spark

tf-serving

The (many) kinds of models you can train

- All your favourite Python libraries* (in Jupyter)
  - Different options to parallelize, with more coming (for now MPI or Beam ish)
- PyTorch
- Tensorflow (along with hyper param tuning with katib)
- mxnet
- etc.

But don't forget about data prep friends!

- ~~For now options are a little limited, but other tools like Apache Spark are in the works~~
  - ~~https://github.com/kubeflow/kubeflow/pull/1467~~
- Now with Spark!*
- Pachyderm
- pandas?
- shell scripts?
- Tensorflow Transform in local mode or GCP Python 2 only.....
- Yeah I guess we did forget about our dataprep friends

*Where now =~ Kubeflow 0.5

Model persistence/deployment/CI

- You need to save your model somewhere
- Your favourite cloud storage provider goes here
  - Why invest in model management when I can make directory?
- ModelDB
- WeaveFlux
- Pachyderm

Model Serving

Python Flask
TensorFlow Model Server
Openvino
NVIDIA Inference Server

Seldon Core

- Routers for A/B tests and multi-armed bandits.
- Supports lots of libraries (Python/Spark/R/etc)
- Monitoring / Security

Or add your own custom magic

How does this fit in to the Ecosystem

Research is great, even better if your boss is OK with just research.

Everyone else has to put models in prod (and also manage models in prod).

Kubeflow bridges the gap from lab to production (with less shell scripts*)

*This is not a guarantee, we're in the EU y'all take that shit seriously.
Also less does not always mean better.

So you want to use this?

What’s Next?!

Step away from keyboard

Think about type(s) of model

Look at components directory and see what’s a fit tool wise

Don’t know? Choose jupyter deal with the details live

Can’t find it?

^^ New Cat Content!!!
^._.^

Getting the chef's recommend pairing:

kfctl.sh* init my_awesome_project --platform {none, gcp, minikube,azure}

cd my_awesome_project

kfctl.sh generate platform && kfctl.sh apply platform

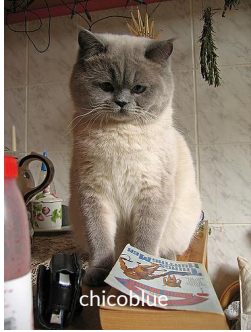
kfctl.sh generate k8s && kfctl.sh apply k8s

Add spark

cd ks_ap && ks pkg install kubeflow/spark

*Use the kfctl binary instead of the shell script for GCP in 0.5+ For IBM see our guide

The chef's recommend pairing is:



- Jupyter Hub
- TF Job & TF Serving
- PyTorch
- Katib (Hyper parameter tuning)
- Ambassador (route requests, access UI)
- Pipelines (Argo + Magic)

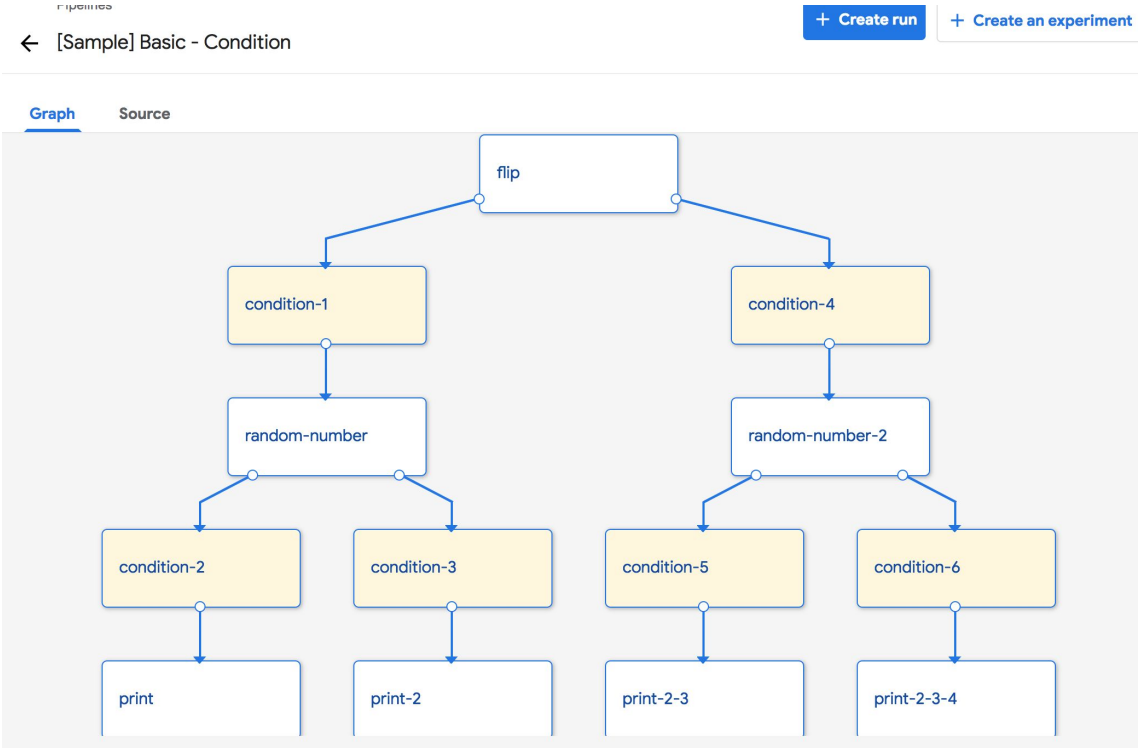
Koziro Hasegawa

What are those pipelines?

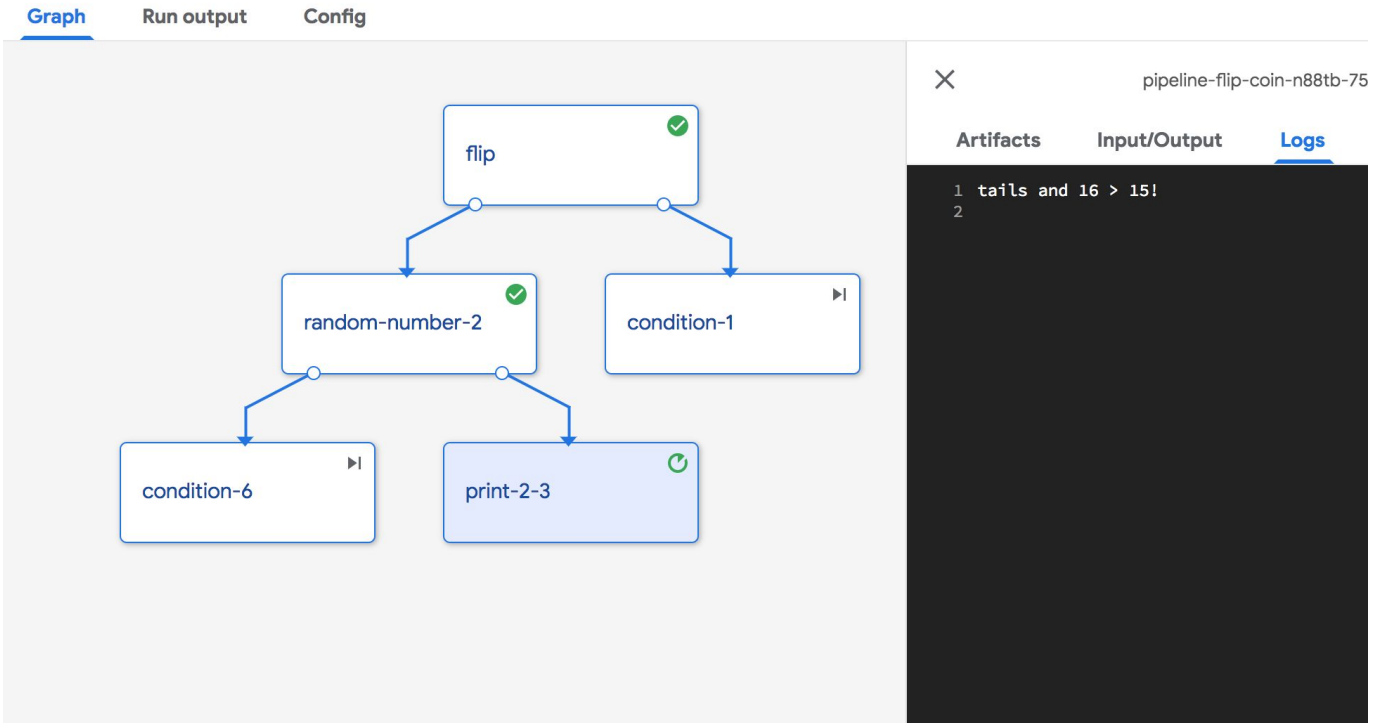
“Kubeflow Pipelines is a platform for building and deploying portable, scalable machine learning (ML) workflows based on Docker containers.” - kubeflow.org

Directed Acyclic Graph (DAG) of “pipeline components” (read “docker containers”) each performing a function.

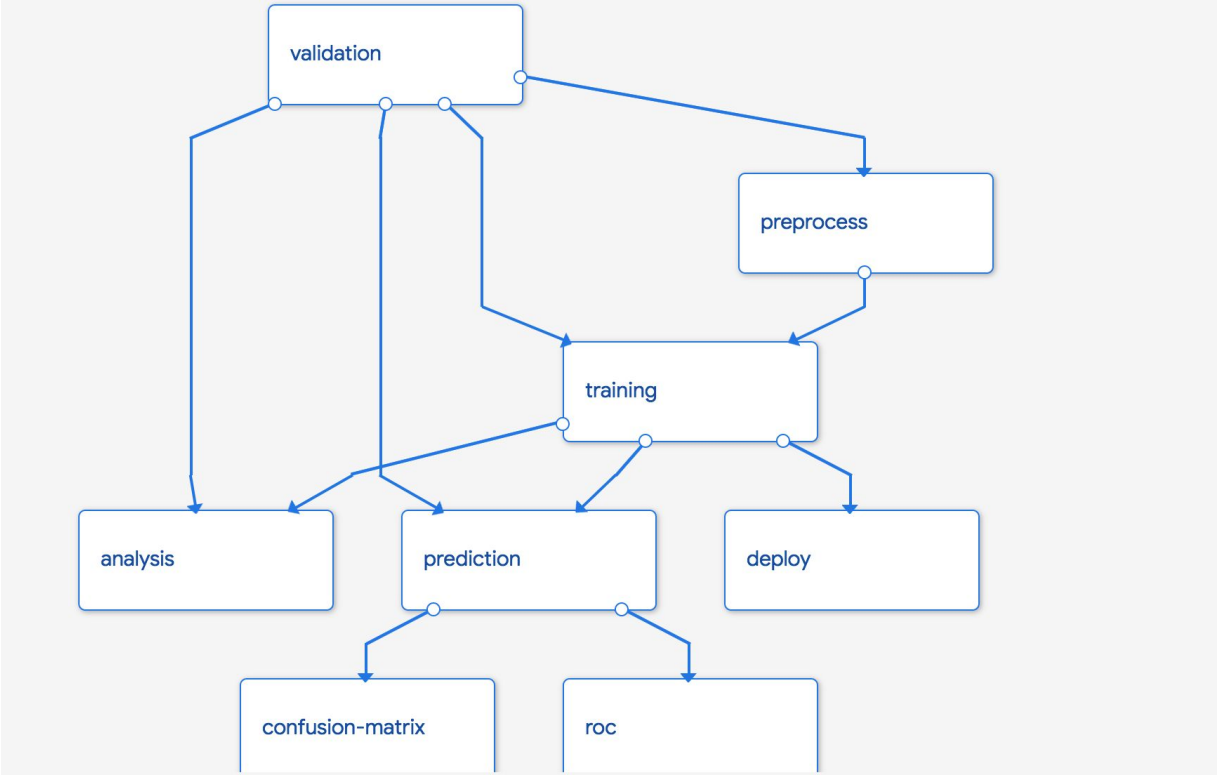
Building that pipeline?



Running that pipeline



Serving that job (not the only way)



Using Spark + TF together in KF

In pure Spark:

- Do data & feature prep using Spark's existing ML APIs or custom logic
- e.g. something like the feature prep in frank

Then either:

- Use TensorFlowOnSpark or similar to skip the copy cost
- Save the results to a PV/DFS/object store & run a seperate TF training job
    - Kind of slow though

A quick detour into PySpark's internals

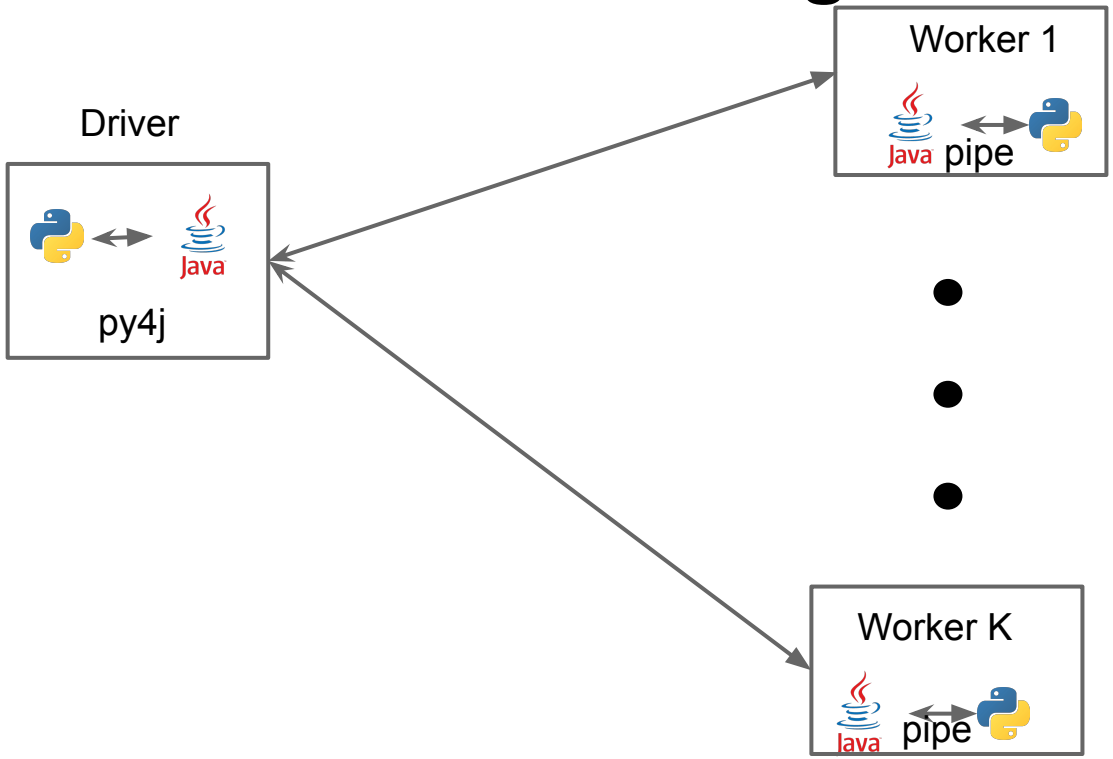
+

+ JSON

PySpark

- The Python interface to Spark
- Same general technique used as the bases for the C#, R, Julia, etc. interfaces to Spark
- Fairly mature, integrates well-ish into the ecosystem, less a Pythonrific API
- Has some serious performance hurdles from the design

So what does that design look like?

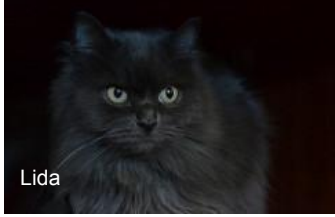


So how does that impact Py[X]

\forall X in {Big Data}-{Native Python Big Data}

- Double serialization cost makes everything more expensive
- Python worker startup takes a bit of extra time
- Python memory isn't controlled by the JVM - easy to go over container limits and not know
- Error messages make ~0 sense
- features aren't automatically exposed, but exposing them is normally simple
- Python dependency management is left to the user or deployment system

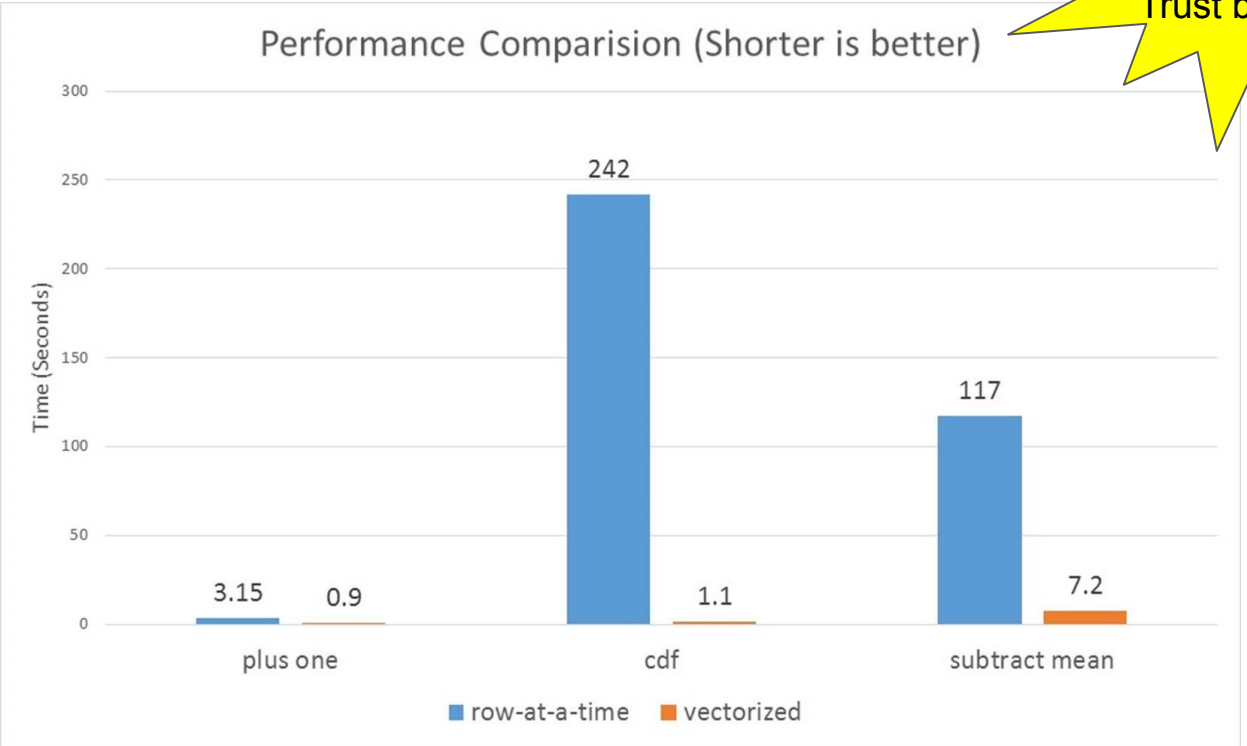
TensorFlowOnSpark, everyone loves mnist!




```
cluster = TFCluster.run(sc, mnist_dist_dataset.map_fun, args,
args.cluster_size, num_ps, args.tensorboard,
TFCluster.InputMode.SPARK)
if args.mode == "train":
  cluster.train(dataRDD, args.epochs)
```

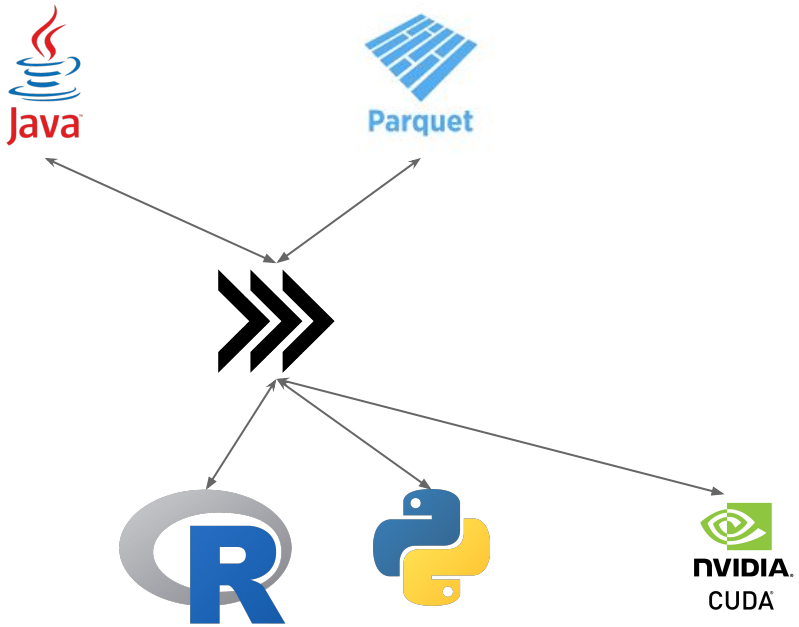
Can we make this better with Arrow?

*Vendor benchmark. Trust but verify.



*Source: https://databricks.com/blog/2017/10/30/introducing-vectorized-udfs-for-pyspark.html.

Arrow (a poorly drawn big data view)



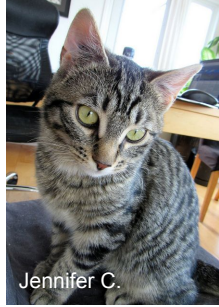
Logos trademarks of their respective projects

How to swap to using Arrow

```
spark.catalog.registerFunction(
    "add", lambda x, y: x + y, IntegerType())
```

=>

```
add = pandas_udf(lambda x, y: x + y, IntegerType())
```



And we can do this in TFOnSpark*:

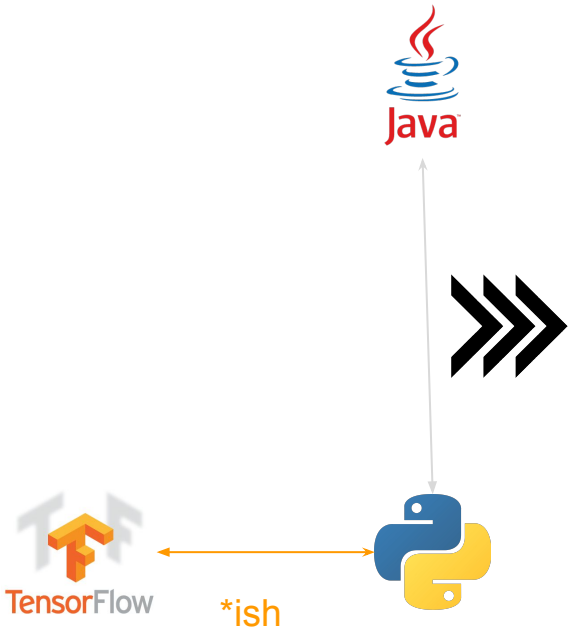

```
unionRDD.foreachPartition(TFSparkNode.train(self.cluster_info,
self.cluster_meta, qname))
```

Will Transform Into something magical (aka fast but unreliable) on the next slide!

Which becomes


```
      train_func = TFSparkNode.train(self.cluster_info,
self.cluster_meta, qname)
      @pandas_udf("int")
      def do_train(inputSeries1, inputSeries2):
        # Sad hack for now
        modified_series = map(lambda x: (x[0], x[1]),
zip(inputSeries1, inputSeries2))
        train_func(modified_series)
        return pandas.Series([0] * len(inputSeries1))
```

And this now looks like:



Logos trademarks of their respective projects

How PySpark Dependencies work:

Python & R folks (or native code human) really painful: especially with TF

K8s:

- Bake your requirements into your container
  - We need better tools because data scientists don't like doing this

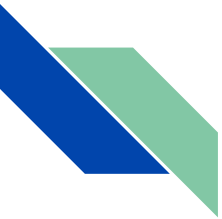
Common YARN Solutions:

- pre-install packages on workers, or runtime install
  - Cloudera*: Give Anaconda some money
  - Google*: I hope you like shell scripts!
  - IBM*: call cloudera, nee hortonworks

What do the rest of the systems do?

- Often very similar designs
- Rarely much better in the general case
- Sometime better integration
  - See Apache Beam and TFX (e.g. TFT/TFMA/etc.)
    - although some other restrictions
- Oooor: save things to disk and call it a day

Cross Cloud: Why?



Cross Cloud OR
The part of the talk where trevo shoot's his employer in the back.

- Clouds are becoming commoditized (thanks K8s)
- This is very good for consumers
- This is very bad for producers (cloud companies)
- You can help! (and it benefits you to do so!)

Clouds are becoming Commoditized

Commoditization

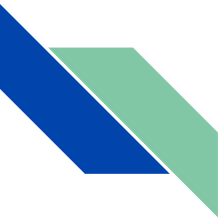
Def: When there is no discernable difference in product

Perfect Competition

Producers must accept market price

MC=MP

Photo Attribution: US Department of Agriculture



Benefit to Consumers (you, if you don't own a Cloud provider)

This is **GREAT** for consumers of Cloud!

K8s on GCP = K8s on IBM = K8s on Azure = K8s on AWS = K8s on _____

In the (near) future, you'll wake up and shift loads based on who is charging least at the moment.

Shortly after that all prices the same.

(Think of Gas stations, gasoline is also a commodity).

Horrible for producers (Cloud companies)

Polar opposite of Vendor Lock-in (there is only one DB2)*

Proprietary gives *limited* monopolistic power.

Can only charge you “going rate” for K8s time, not “whatever we want and fu” rate.

*I’m using DB2 as example bc I work for IBM so it’s safe, but I’m looking at you too Oracle, AWS, Terra Data, Micro$oft, GCP, blah blah blah everyone.

And you can help make this happen faster!

How to help:

Open enterprise accounts w all K8s producers (or at least a few).

Make apps that you can shift between providers.

Throw a hissy fit at sales when moving apps doesn't work right (it's at least one cloud providers fault and maybe both).

Shift loads based on cost of the day.

Cross Cloud: How?

What does "cross-cloud" mean here

- Train on multiple clouds/on-prem at the same time with a unified result
  - Very hard, limited benefit, technically possible if you buy me a drink
- Train on one cloud deploy on serve another cloud(s)
  - Hacky, but not that hard, possibly useful
- Portable pipelines (e.g. same pipeline on multiple clouds)
  - Aww yeah party city*, possibly useful

Train on cheapest cloud of the day

- Expensive GPUs vs cheap CPUs
- Free cycles on-prem?
- Put all your “favorite” cloud providers in a knife fight to the death for your ~~love~~ dollars
- "Data gravity" may impact this

Train on cloud X; serve on cloud Y

Just because things were cheap on X back when you trained on Tuesday, doesn’t mean you can’t shift workloads to Y.

Training rigs cost different than serving rigs (e.g. GPUs vs CPUs). Train where its cheap to train, serve where its cheap or best to serve.

"Sensitive" Data? Train on Prem- serve in cloud

Portable Pipelines

Run experiment `super-secret-squirrel-5a1sg`on different clouds and get the same *reproducible results*.

Ahh reproducible results, you put the “science” back in “data science”.

But really, this is important.

Also **data gravity and how to fight it.** (how to buy your way out of it on the cheap).

Making that pipeline cross-cloud

Many example KF pipelines use persistent volumes to store models and other outputs

To allow our pipeline's data to go across clouds we can either re-architect our tools to use an object store or add an additional pipeline stage a-la "hacky-s3-copy_final_2.py.sh"*

Run KF on cloud 2 and "hacky-s3-downloader.py" the result into a local PV and roll from there

Serve it with Seldon!**

*Only sort of joking
** Thanks for the tweet :p

Live streamed demos (recorded on YouTube)

- Kubeflow intro https://codelabs.developers.google.com/codelabs/kubeflow-introduction/index.html & streamed http://bit.ly/kfIntroStream
- Kubeflow E2E with Github issue summurizationhttps://codelabs.developers.google.com/codelabs/cloud-kubeflow-e2e-gis/ & streamed http://bit.ly/kfGHStream
- DIY Multi-cloud workshop https://github.com/intro-to-ml-with-kubeflow/intro-to-ml-with-kubeflow-examples/tree/master/multi-cloud & streamed http://bit.ly/kf_multi_demo_video
- You can tell they were live streamed by how poorly went, I promise no video editing has occurred.

Why you shouldn't use this?

Downsides to Kubeflow

- Lot's of overhead versus doing it locally
- Active development (look it's 0.5)
  - The web UI changed colours recently
  - The startup scripts are being re-written from sh to go
  - Ksonnet is being replaced with
- 3 talks on Kubeflow can give you 3 different toolsets

Very DIY KF Cross Cloud Workshop

Workshop instructions:

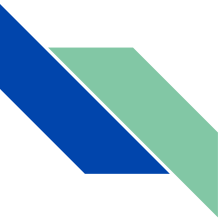
http://bit.ly/kfCrossCloudWorkshop

Workshop slides:

http://bit.ly/kfSlides3

Workshop discussion doc:

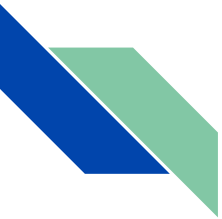
http://bit.ly/kfCrossCloudDoc



Kubeflow book!

http://www.introductiontomlwithkubeflow.com/

Questions? Half-baked Demo?



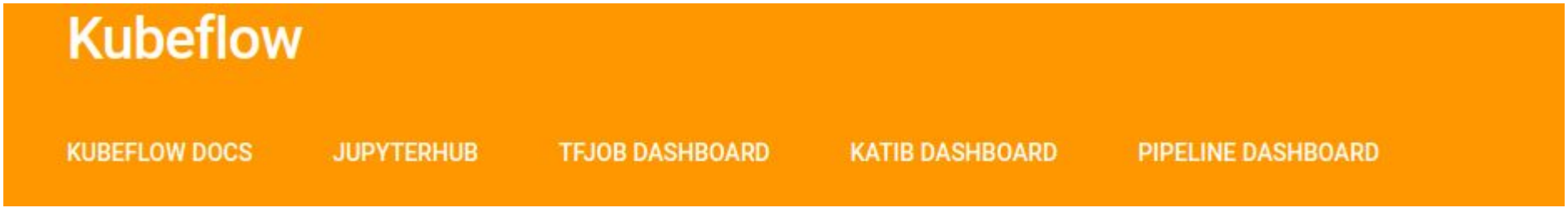
Kubeflow book!

http://www.introductiontomlwithkubeflow.com/

Connect to the Kubeflow Web UI

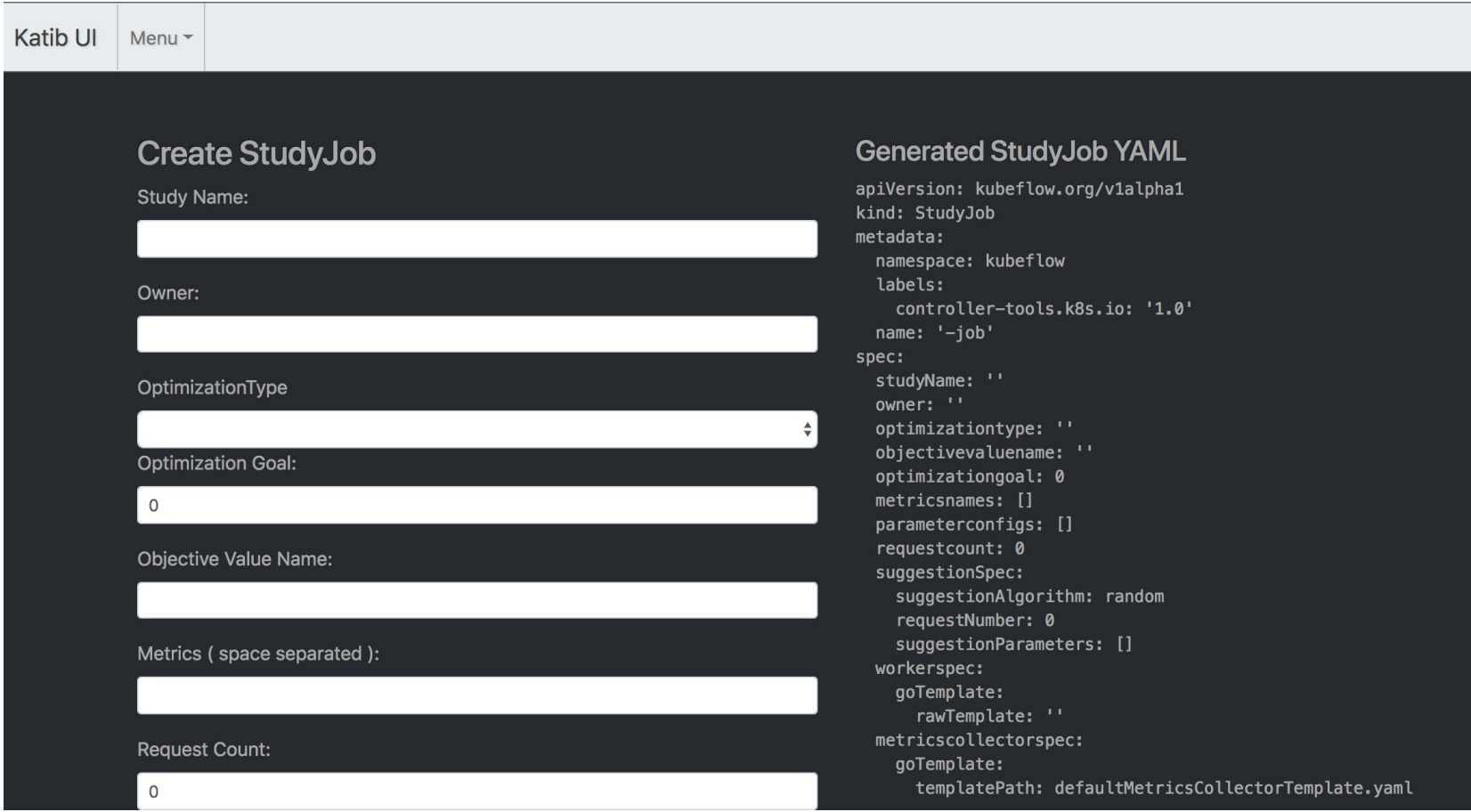
kubectl port-forward svc/ambassador -n kubeflow 8080:80 &

Or use IAP, but that's… another story



When you don't know anything or know a lot about ML: Hyper Parameter Tuning

- Katib
  - Does not depend on a specific ML tool (e.g. not just TF)
  - Supports a few different search algorithms
  - e.g. "What should I set my L1 regularization too? Idk let's ask the computer"
- Great way to accidently overfit too!* (*if you're not careful)
- As respective cloud providers, we are happy to rent you a lot of resources
- Seriously, mention our names in the sales call. We're both going for promo (and that shit is hard)


Katib UI
Menu
Create StudyJob
Study Name:
Owner:
OptimizationType
Optimization Goal:
0
Objective Value Name:
Metrics (space separated):
Request Count:
0
Generated StudyJob YAML
apiVersion: kubeflow.org/v1alpha1
kind: StudyJob
metadata:
namespace: kubeflow
labels:
controller-tools.k8s.io: '1.0'
name: '-job'
spec:
studyName: ''
owner: ''
optimizationtype: ''
objectivevaluename: ''
optimizationgoal: 0
metricsnames: []
parameterconfigs: []
requestcount: 0
suggestionSpec:
suggestionAlgorithm: random
requestNumber: 0
suggestionParameters: []
workerspec:
goTemplate:
rawTemplate: ''
metricscollectorspec:
goTemplate:
templatePath: defaultMetricsCollectorTemplate.yaml

But what about [special foo-baz-inator] or [special-yak-shaving-tool]?

Write a Dockerfile and build an image, use FROM so you're not starting from scratch.

```
FROM gcr.io/kubeflow-images-public/tensorflow-1.6.0-notebook-cpu

RUN pip install py-special-yak-shaving-tool
```

Then tell set it as a param for your training/serving job as needed:

```
ks param set  tfjob-v1alpha2 image "my-special-image-goes-here"
```

Now your fortran lives forever!