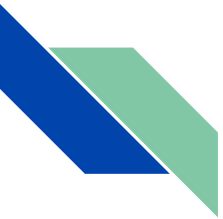
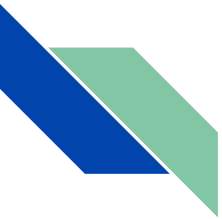


Thanks for coming early to **Introducing KF with Apache Spark & TF**:
If you're bored you can do this **mostly unrelated** survey:

http://bit.ly/holdenTestingSpark

Introducing Kubeflow & w/ Apache Spark & TF

With your friend @holdenkarau
May contains references to Cthulhu & Hannah Montana
Does contain pictures of cats & dogs
No endorsement implied by my employer, Cthulhu, Hannah Montana, cats, or dogs



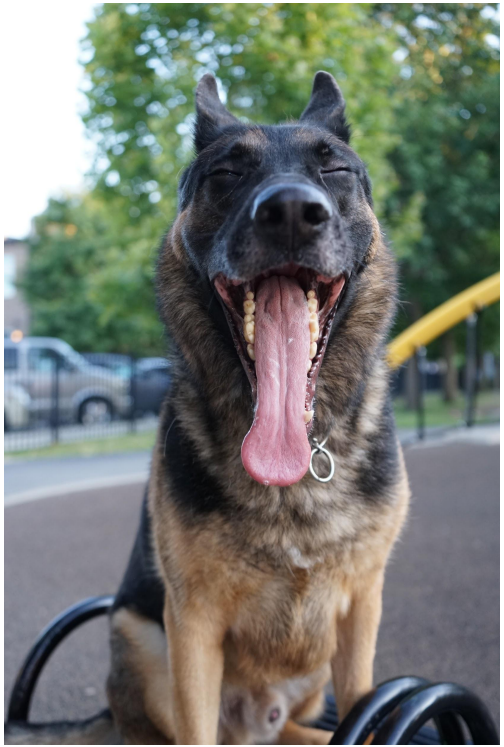
Agenda

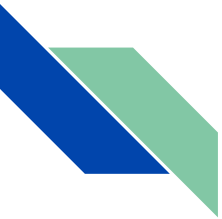
- About Me
- Why Trevor isn't here
- What Is KubeFlow?
- How to build a pipeline with Kubeflow, Spark, and friends
- Validating your ML pipelines (regardless of KF)
- Link some examples / workshops
- Link to videos of me doing the workshops poorly

Holden About Me Slides

- Prefered pronouns are she/her
- Apache Spark PMC / ASF member + contributor on lots of other projects
- previously IBM, Alpine, Databricks, Google, Foursquare & Amazon
- co-author of Learning Spark & High Performance Spark
- Twitter: @holdenkarau
- Slide share http://www.slideshare.net/hkarau
- Code review livestreams: https://www.twitch.tv/holdenkarau / https://www.youtube.com/user/holdenkarau
- Talk Videos http://bit.ly/holdenSparkVideos
- Talk feedback: http://bit.ly/holdenTalkFeedback

Why Trevor isn't here





Open Source Kubeflow Salesman:

Open Source Kubeflow Salesman: **Slaps roof of Kubeflow**

THIS BAD BOY CAN FIT SO MANY BUZZWORDS IN IT

Image from slon

Kubeflow

Open Source Kubeflow Salesman: **Slaps roof of Kubeflow**

THIS BAD BOY CAN FIT SO MANY BUZZWORDS IN IT

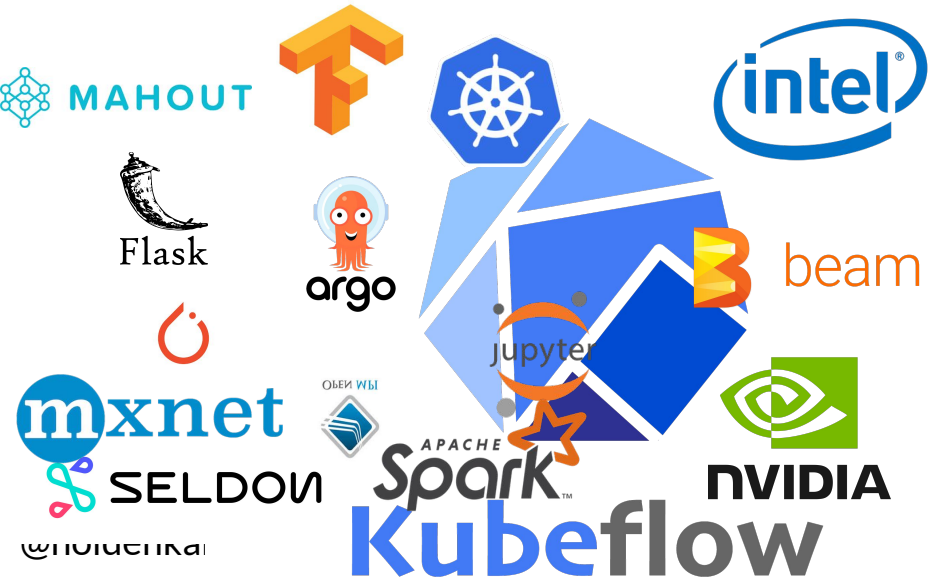



Image from slon

Background

Things I thought you might want a refresher on

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)

+

What is Kubernetes?

+

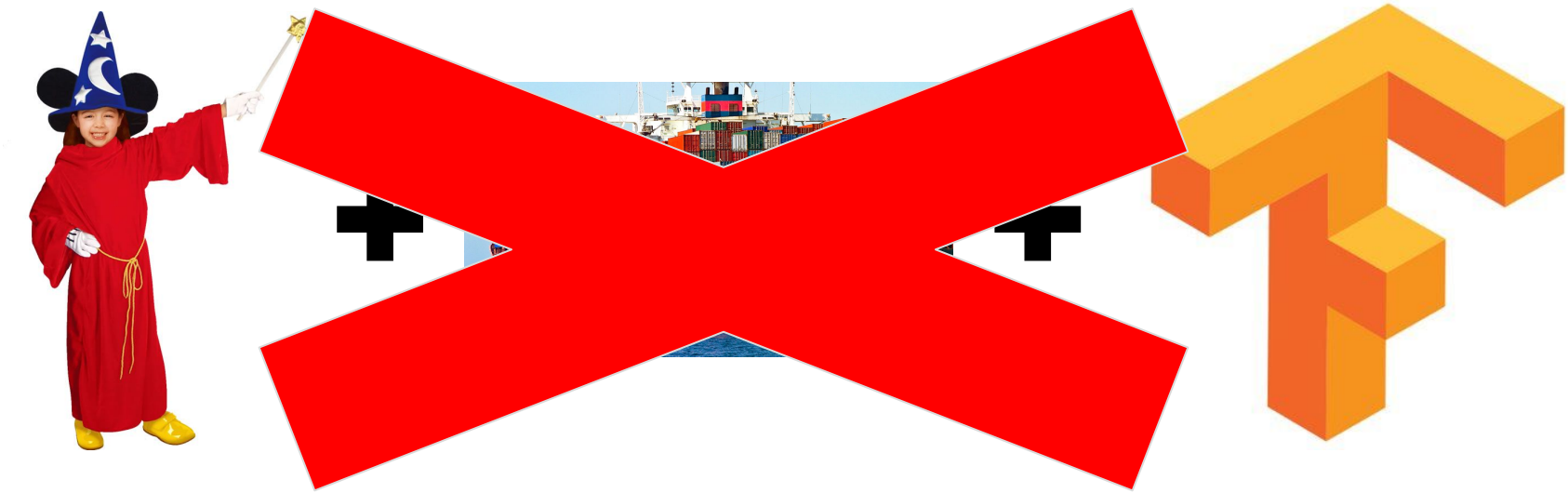
So what is Kubeflow?

What is Kubeflow?

+

+

What is Kubeflow?



What is Kubeflow?

What is Kubeflow?

Photo from Roland Tanglao / SnoZu

Stone Soup is an old folk story in which hungry strangers convince the people of a town to each share a small amount of their food in order to make a meal that everyone enjoys, and exists as a moral regarding the value of sharing.

- https://en.wikipedia.org/wiki/Stone_Soup

Components Buffet*

0.6*: https://github.com/kubeflow/manifests

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.

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:
0.6: Replace kfctl.sh w/kfctl and no ks_app or ks

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

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

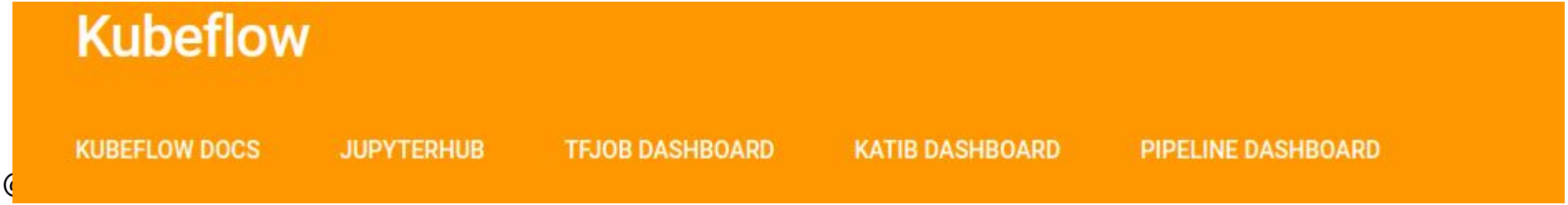
Connect to the Kubeflow Web UI

0.6: istio + envoy instead

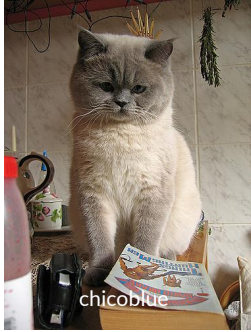
kubectl port-forward svc/ambassador -n kubeflow 8080:80 &

Or use IAP, but that's… another story

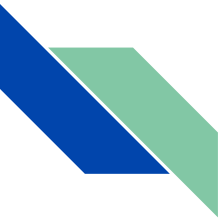
The UI changed between when I made this slide and today: it's now blue



The chef's recommend pairing is:



- Jupyter Hub
- TF Job & TF Serving
- PyTorch
- Katib (Hyper parameter tuning)
- Ambassador (makes it easier to access the UIs)
- Pipelines (Argo + Magic)

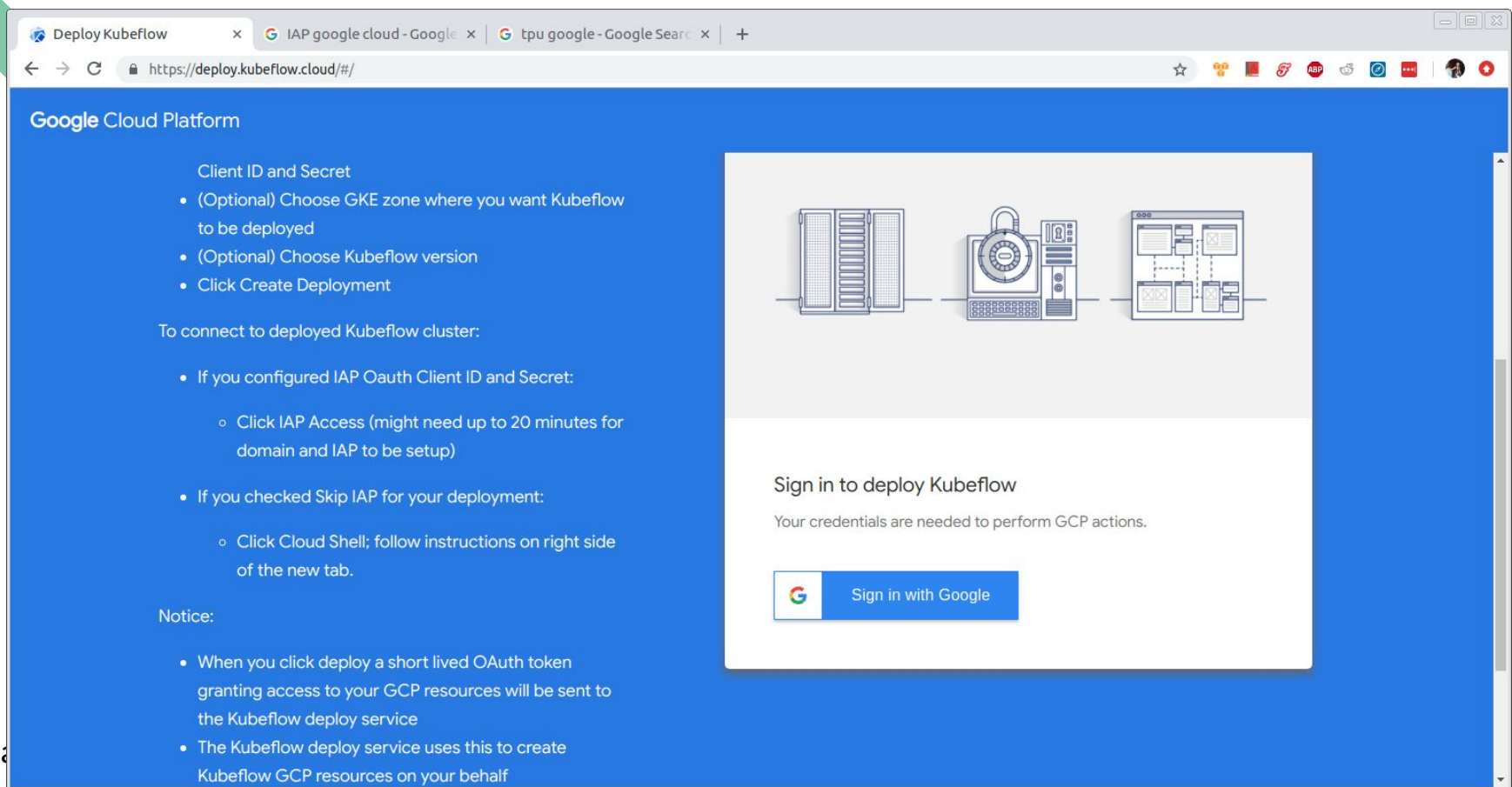


Click-to-deploy: get started hella fast* on GCP

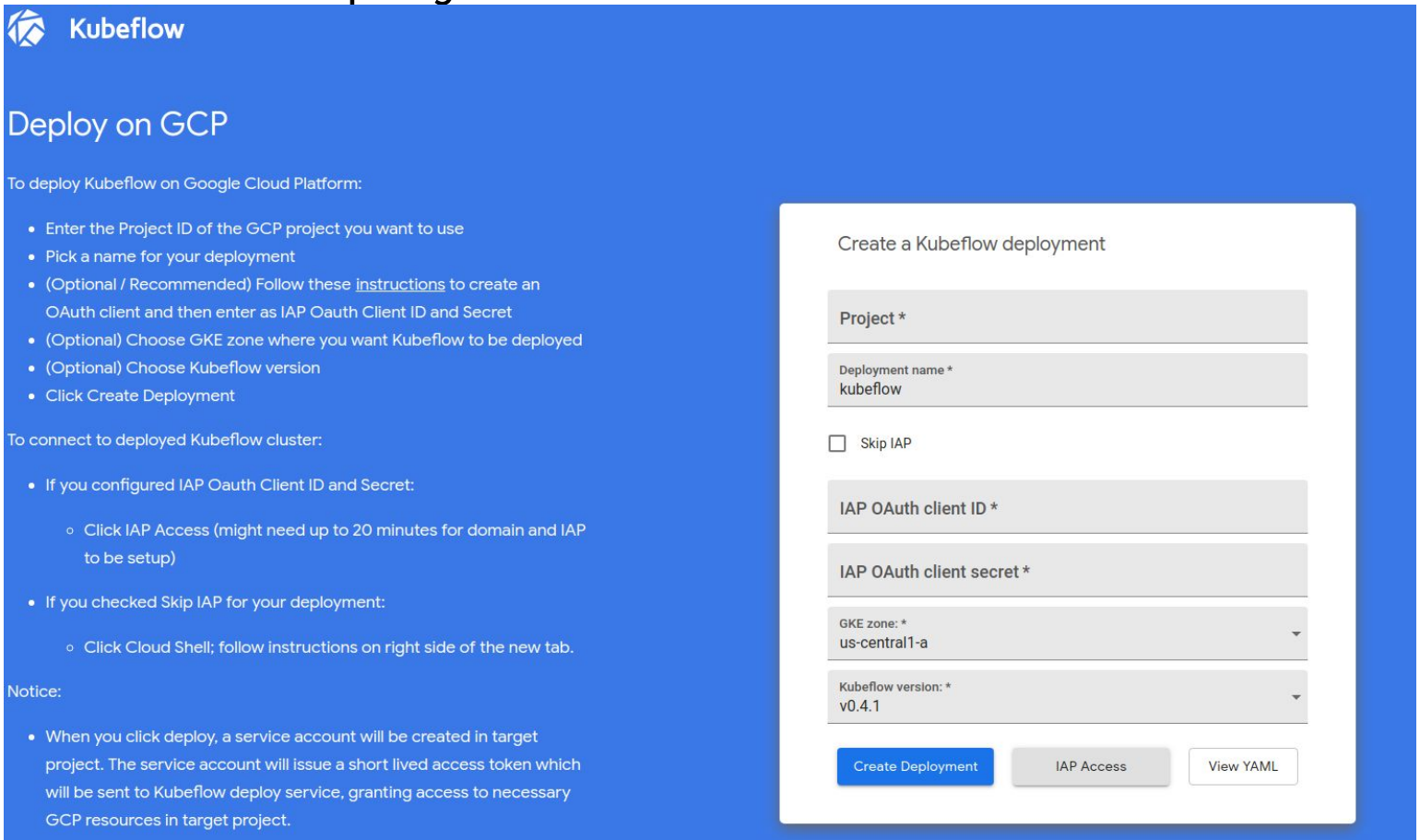
https://deploy.kubeflow.cloud

Not the only way, just hella fast

Click-to-deploy continued

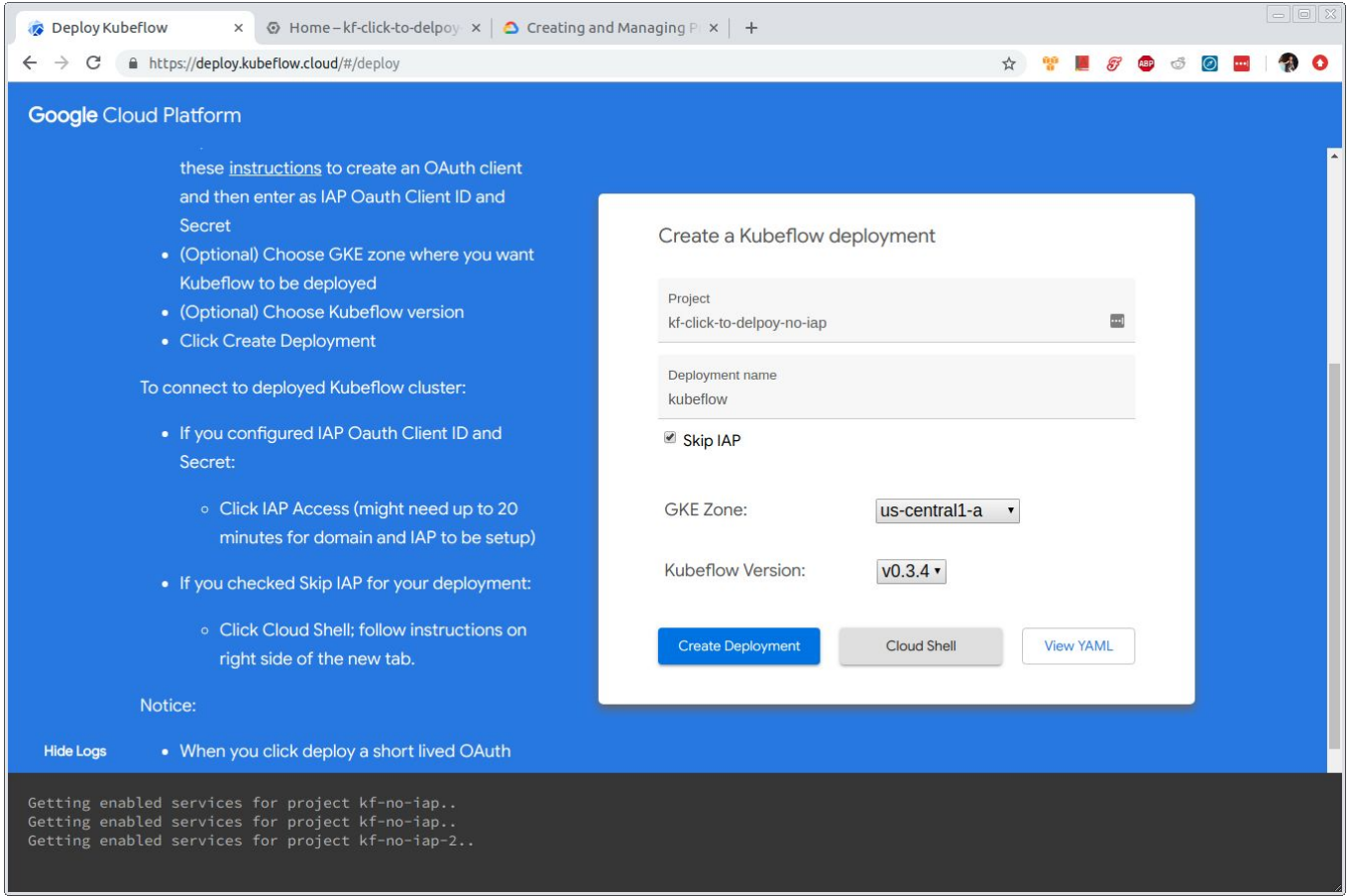


Click-to-deploy continued



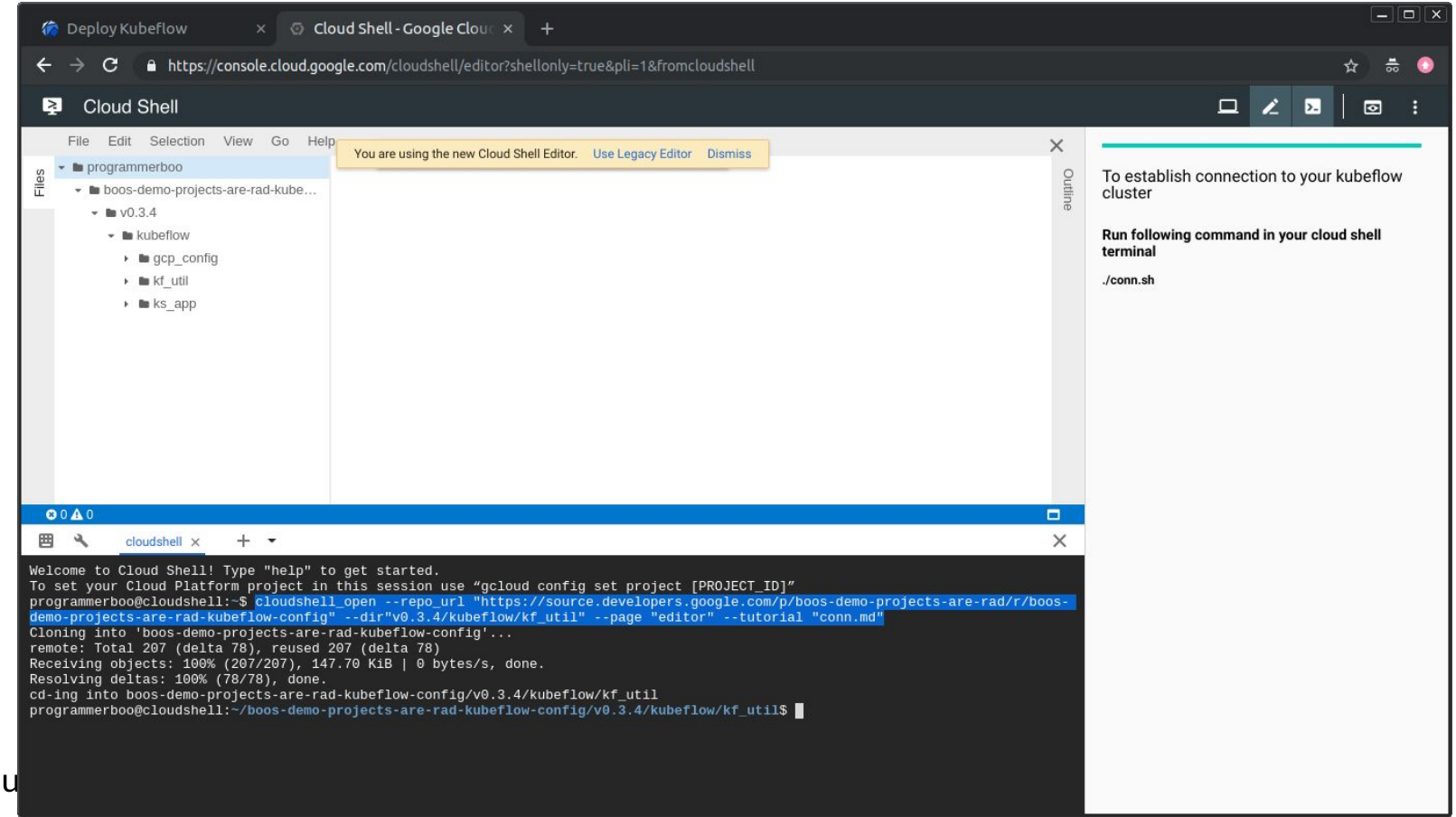
@holdenkarau

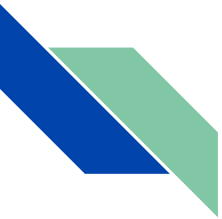
Click-to-deploy continued



@holdenkarau

Click-to-deploy continued



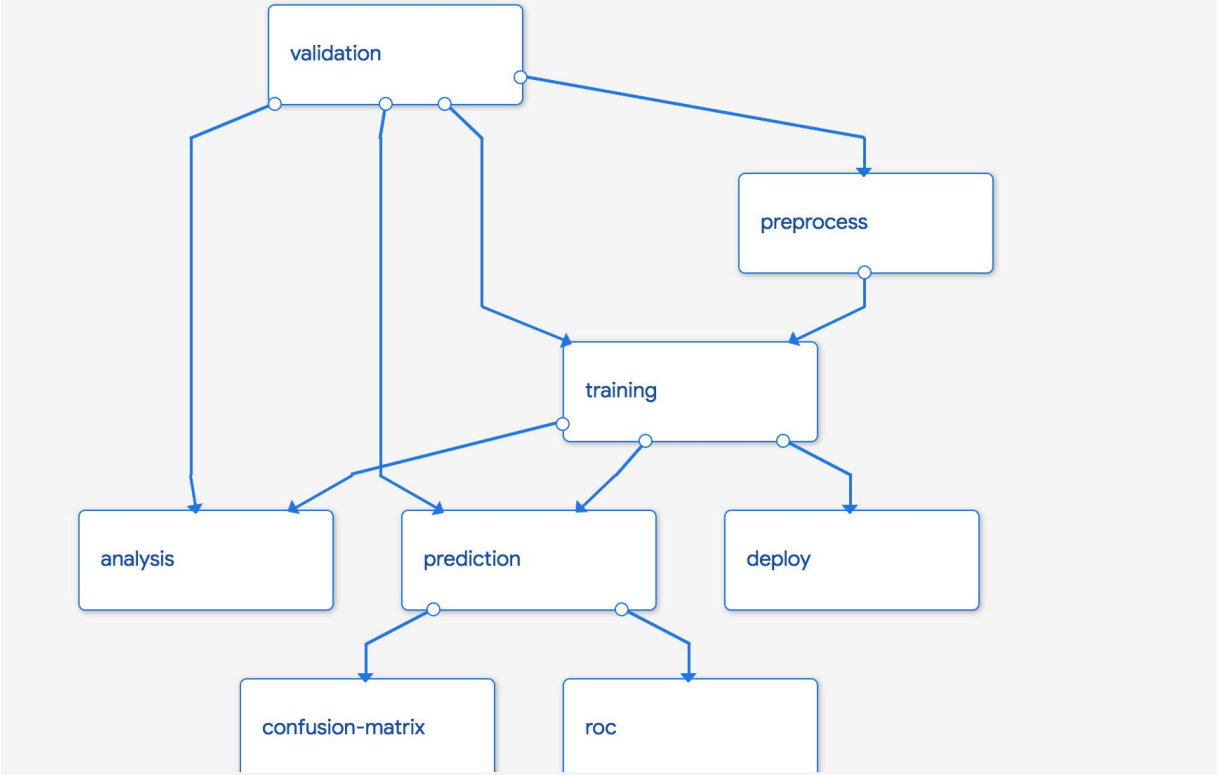


What are those pipelines?

“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.

Serving that job (not the only way)



Specifying a pipeline in 0.6+

```
@dsl.pipeline(
    name='Simple sci-kit KF Pipeline',
    description='A simple end to end sci-kit seldon kf pipeline'
)
def mnist_train_pipeline(
        docker_org="index.docker.io/seldonio",
        train_container_version="0.2",
        serve_container_version="0.1"):
```

Creating volumes:

```
vop = dsl.VolumeOp(
     name="create_pvc",
     resource_name="nfs-1",
     modes=dsl.VOLUME_MODE_RWO,
     size="10G")
 volume = vop.volume
```

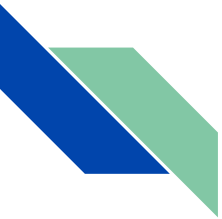
Running arbitrary training code:

There are lots of other ways to do this, including generating from a notebook

```
train = dsl.ContainerOp(
        name='sk-train',
        image=f"{docker_org}/skmnistclassifier_trainer:{train_container_version}",
        pvolumes={"/data": volume})
```

Submitting a TF job:

```
   tfjobjson_template = Template("""
{
        "apiVersion": "kubeflow.org/v1beta1",
        "kind": "TFJob",
        "metadata": {
                "name": "mnist-train-{{workflow.uid}}",
                "ownerReferences": [
                {
                        "apiVersion": "argoproj.io/v1alpha1",
                        "kind": "Workflow",
                        "controller": true,
                        "name": "{{workflow.name}}",
                        "uid": "{{workflow.uid}}"
...
```



Kubeflow pipeline info:

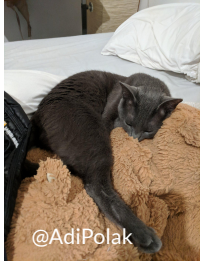
https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/

https://www.kubeflow.org/docs/pipelines/pipelines-quickstart/

https://github.com/kubeflow/pipelines/tree/master/samples

How do we use this with Spark?*

Install Spark into Kubeflow 0.5



ks pkg install kubeflow/spark

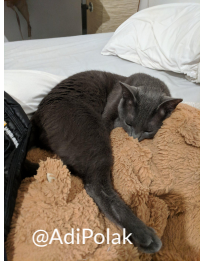
ks generate spark-operator spark-operator --name=spark-operator

ks apply default -c spark-operator

Submit a job 0.5

```
# Create a Spark job with the operator (Pi)

ks generate spark-job spark-pi --name=spark-operator \


--applicationResource="local:///opt/spark/examples/jars/spark-examples_2.11-2.3.1.jar" \

  --mainClass=org.apache.spark.examples.SparkPi

ks apply default -c spark-pi
```

Install Spark into Kubeflow 0.6



Edit the template (
https://raw.githubusercontent.com/kubeflow/kubeflow/v0.6-branch/bootstrap/config/kfctl_k8s_istio.0.6.2.yaml
) to point to my manifest Spark PR (https://github.com/kubeflow/manifests/pull/174) and add Spark to the app.yaml

Note: please don't actually do this, wait until PR 174 is merged

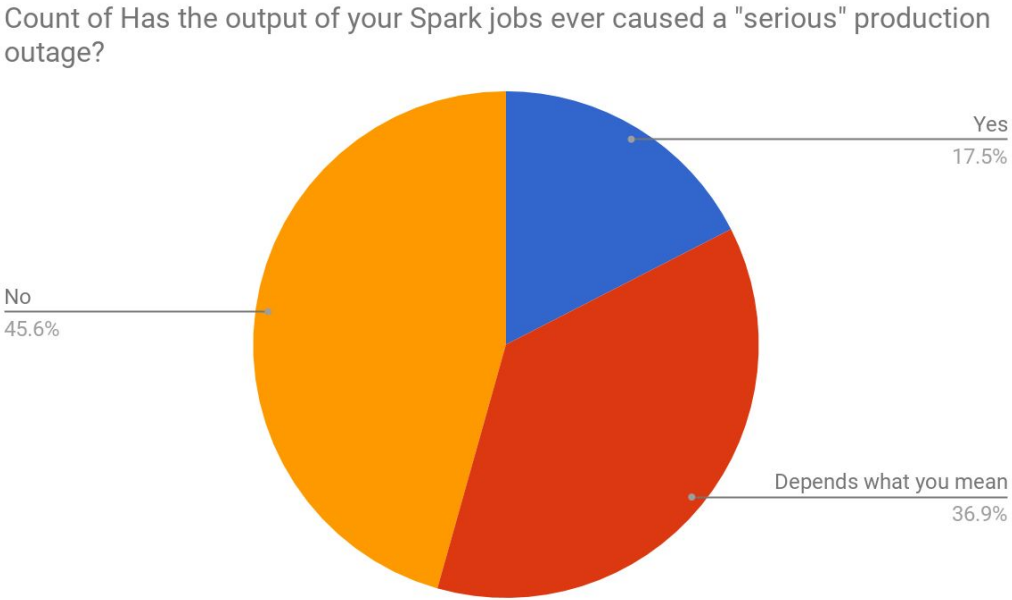
Why you shouldn't use Kubeflow?

Downsides to Kubeflow

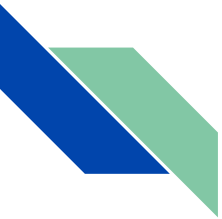
- Lot's of overhead versus doing it locally
- Active development (look it's 0.6)
  - A lot of key components are going to change really really soon
  - kfctl, ksonnet, etc.
  - This talk only works in 0.4 & 0.5, I need to re-add Spark in 0.6+
- 3 talks on Kubeflow can give you 3 different toolsets

How do we keep from destroying the world?

Why you need to validate:



Results from: Testing with Spark survey http://bit.ly/holdenTestingSpark



Pipeline & Model Validation

- Just because it worked once doesn't mean it will always work
- At some point you will have to update your models
- Even if you use a pipeline (please do) so it's repeatable the world around you may change
    - There are lots of funny/sad stories that go here

So how do we validate our jobs?

- The idea is, at some point, you made software which worked.
  - If you don't you probably want to run it a few times and manually validate it
- Maybe you manually tested and sampled your results
- Hopefully you did a lot of other checks too
- But we can't do that every time, our pipelines are no longer write-once run-once they are often write-once, run forever, and debug-forever.

General Rules for making Validation rules

- According to a sad survey most people check execution time & record count
- spark-validator is still in early stages but interesting proof of concept
  - I was probably a bit sleep deprived when I wrote it because looking at it… idk
  - I have a rewrite which is going through our open source releasing process. Maybe it will be released! Not a guarantee.
- Sometimes your rules will miss-fire and you'll need to manually approve a job
- Historical data
- Domain specific solutions
- **Do you have property tests?**

% of data change

- Not just invalid records, if a field's value changes everywhere it could still be "valid" but have a different meaning
  - Remember that example about almost recommending illegal content?
- Join and see number of rows different on each side
- Expensive operation, but if your data changes slowly / at a constant ish rate
  - Sometimes done as a separate parallel job
- Can also be used on output if applicable
  - You do have a table/file/as applicable to roll back to right?

Validation rules can be a separate stage(s)

- Sometimes data validation in parallel in a separate process
- Combined with counters/metrics from your job
- Can then be compared with a seperate job that looks at the results and decides if the pipeline should continue

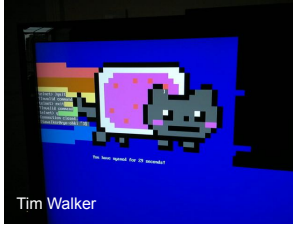
TFDV: Magic*

Counters, schema inference, anomaly detection, oh my!

```
# Compute statistics over a new set of data

new_stats = tfdv.generate_statistics_from_csv(NEW_DATA)

# Compare how new data conforms to the schema

anomalies = tfdv.validate_statistics(new_stats, schema)

# Display anomalies inline

tfdv.display_anomalies(anomalies)
```

Details:
https://medium.com/tensorflow/introducing-tensorflow-data-validation-data-understanding-validation-and-monitoring-at-scale-d38e3952c2f0

What can we learn from TFDV:



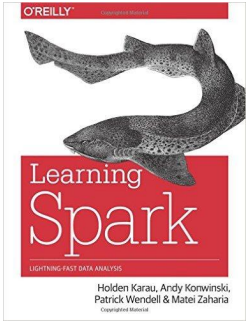
- Auto Schema Generation & Comparison
  - Spark SQL yay!
- We can compute summary statistics of your inputs & outputs
  - Spark SQL yay!
- If they change a lot "something" is on fire
- Anomaly detection: a few different spark libraries & talks here
  - Can help show you what might have gone wrong

Not just data changes: Software!

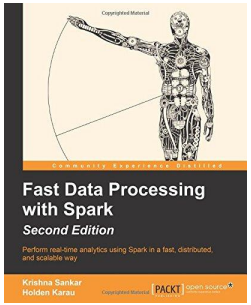
- Things change! Yay! Often for the better.
  - Especially with handling edge cases like NA fields
  - Don't expect the results to change - side-by-side run + diff
- Excellent PyData London talk about how this can impact ML models
  - Done with sklearn shows vast differences in CVE results only changing the model number

Live streamed demos (recorded on YouTube)

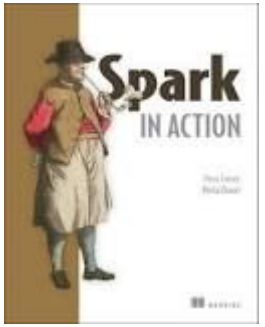
- Kubeflow intro https://codelabs.developers.google.com/codelabs/kubeflow-introduction/index.html & streamed http://bit.ly/kfIntroStream
- Kubeflow E2E with Github issue summurizationhttps://codelabs.developers.google.com/codelabs/cloud-kubeflow-e2e-gis/ & streamed http://bit.ly/kfGHStream
- And more on https://youtube.com/user/holdenkarau
- You can tell they were live streamed by how poorly went, I promise no video editing has occurred.



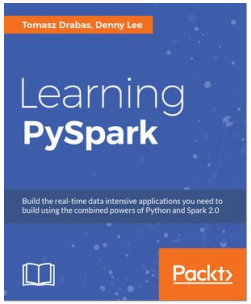
Learning Spark



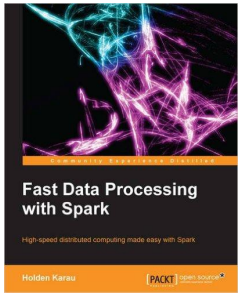
Fast Data Processing with Spark (2nd edition)



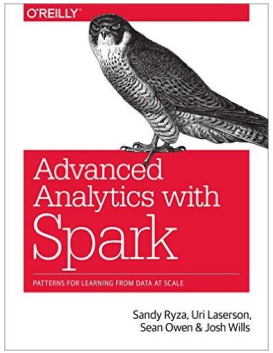
Spark in Action



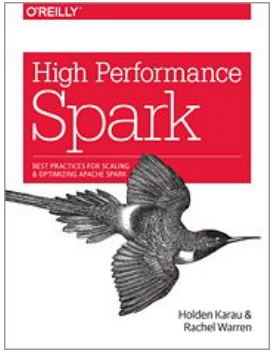
Learning PySpark



Fast Data Processing with Spark (Out of Date)

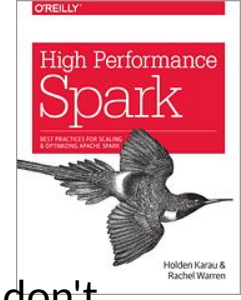


Advanced Analytics with Spark



High Performance Spark

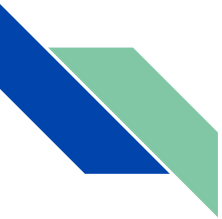
High Performance Spark!



Available today, nothing related to this presentation, but you should still buy it. If you don't want to buy it there is a free book signing @ ~3:15pm @ the ORM booth.

Cat's love it!

Amazon sells it: http://bit.ly/hkHighPerfSpark :D



A book "soon"* on ML w/ Kubeflow

introductiontomlwithkubeflow.com

*For values of soon which the publisher & myself may have different understandings


Distributed
Computing
4 Kids
Apache Spark
Sign up for the mailing list @
http://distributedcomputing4kids.com

Counters* to the rescue**!

- Both BEAM & Spark have their it own counters
  - Per-stage bytes r/w, shuffle r/w, record r/w. execution time, etc.
  - In UI can also register a listener from spark validator project
- We can add counters for things we care about
  - invalid records, users with no recommendations, etc.
  - Accumulators have some challenges (see SPARK-12469 for progress) but are an interesting option
- We can _pretend_ we still have nice functional code

*Counters are your friends, but the kind of friends who steal your lunch money

** In a similar way to how regular expressions can solve problems….

So what does that look like?


```
val parsed = data.flatMap(x => try {
    Some(parse(x))
    happyCounter.add(1)
  } catch {
    case _ =>
      sadCounter.add(1)
      None // What's it's JSON
  }
}
// Special business data logic (aka wordcount)
// Much much later* business error logic goes here
```

Pager photo by Vitachao CC-SA 3

Ok but what about those *s

- Beam counters are implementation dependent
- Spark counters aren’t great for data properties
- etc.

Extra considerations for ML jobs:

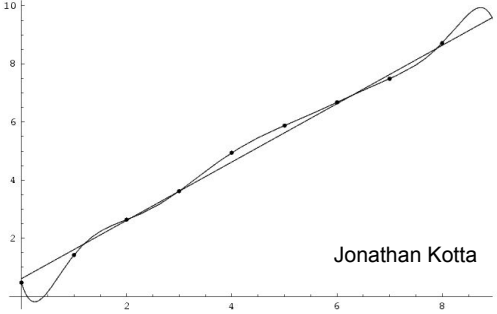
- Harder to look at output size and say if its good
- We can look at the cross-validation performance
- Fixed test set performance
- Number of iterations / convergence rate
- Number of features selected / number of features changed in selection
- (If applicable) \delta in model weights or \delta in hyper params

Updating your model

- The real world changes
- Online learning (streaming) is super cool, but hard to version
  - Common kappa-like arch and then revert to checkpoint
  - Slowly degrading models, oh my!
- Iterative batches: automatically train on new data, deploy model, and A/B test
- But A/B testing isn't enough -- bad data can result in wrong or even illegal results

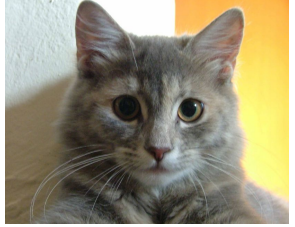
Cross-validation

because saving a test set is effort

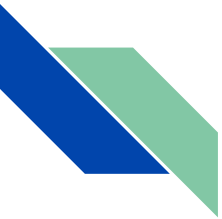


- Trains on X% of the data and tests on Y%
  - Multiple times switching the samples
- Can do hyper parameter tuning
  - Kubeflow has katib
  - org.apache.spark.ml.tuning has the tools for auto fitting using CV
  - If your going to use this for auto-tuning please please save a test set
  - Otherwise your models will look awesome and perform like a ford pinto (or whatever a crappy car is here. Maybe a renault reliant? Or an especially crap Holden car)

False sense of security:



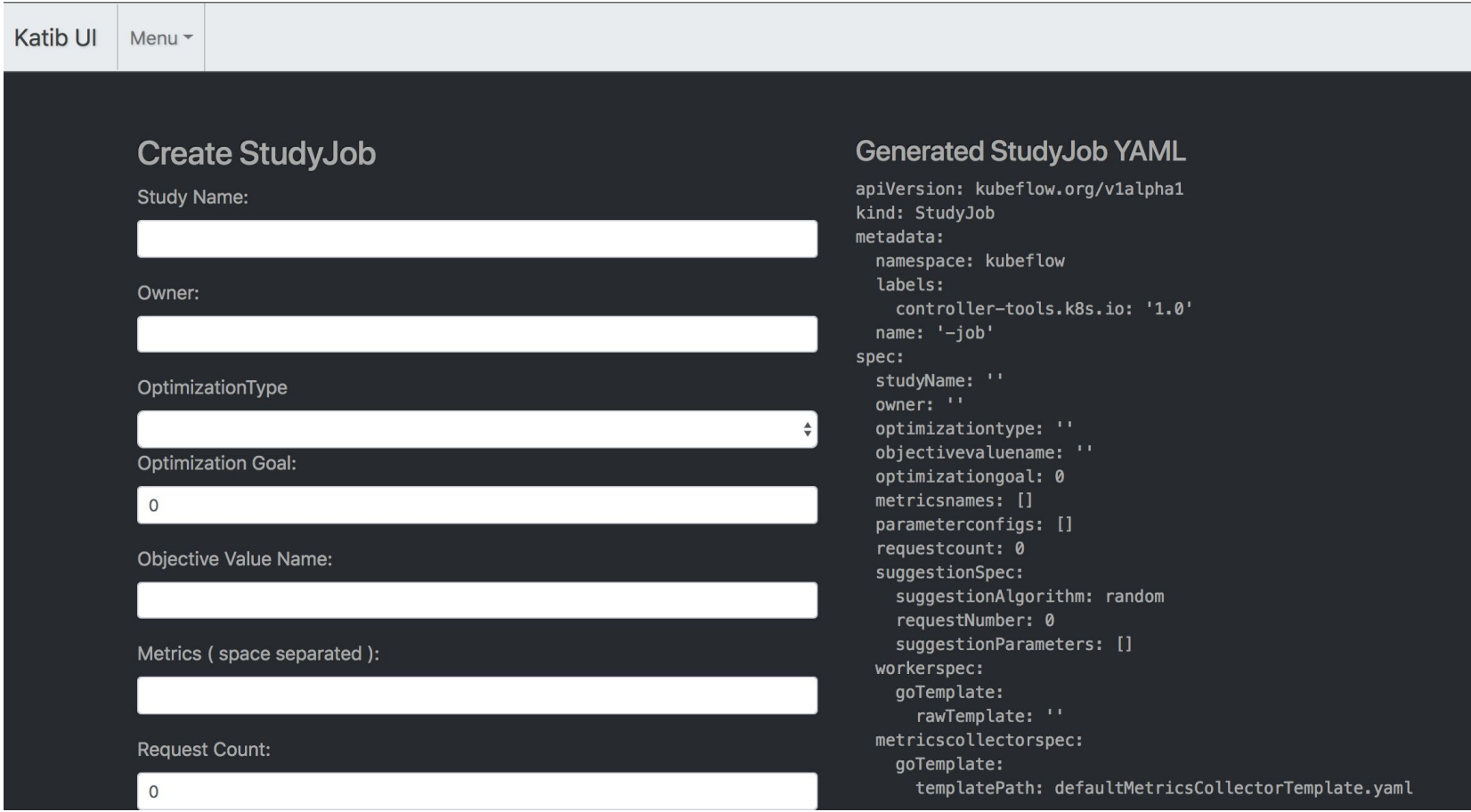
- A/B test please even if CV says amazing
- Rank based things can have training bias with previous orders
  - Non-displayed options: unlikely to be chosen
  - Sometimes can find previous formulaic corrections
  - Sometimes we can “experimentally” determine
- Other times we just hope it’s better than nothing
- Try and make sure your ML isn’t evil or re-encoding human biases but stronger

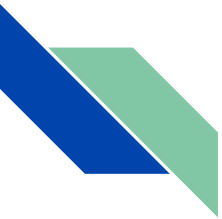


Questions? Free signed books (@ 3:15pm)

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!