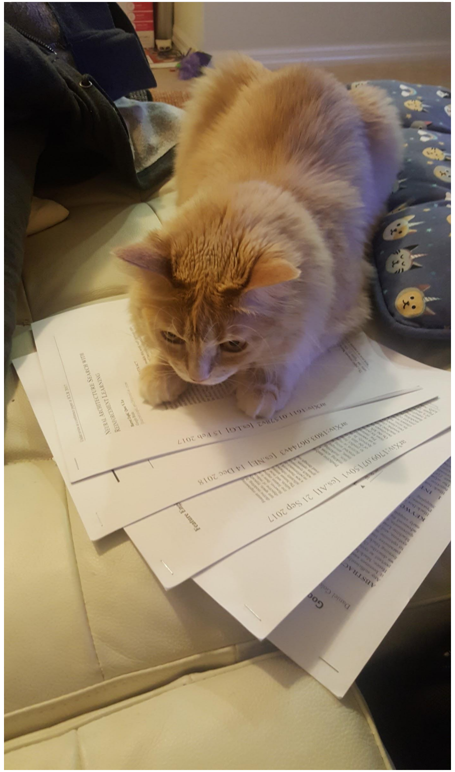
KubeCon | CloudNativeCon

North America 2019

Measuring and Optimizing Kubernetes Usage at Lyft

Richard Liu, Senior SWE @ Google
Konstantin Gizdarski, SWE @ Lyft

Who are we?



What will you learn?

Lyft and Cloud Infrastructure Spending

Shipping Infraspend 2.0

Machine Learning Platform at Lyft

Extending Infraspend 2.0 to support Multi-tenant Platforms

Ingesting and Presenting Kubernetes Data in Infraspend

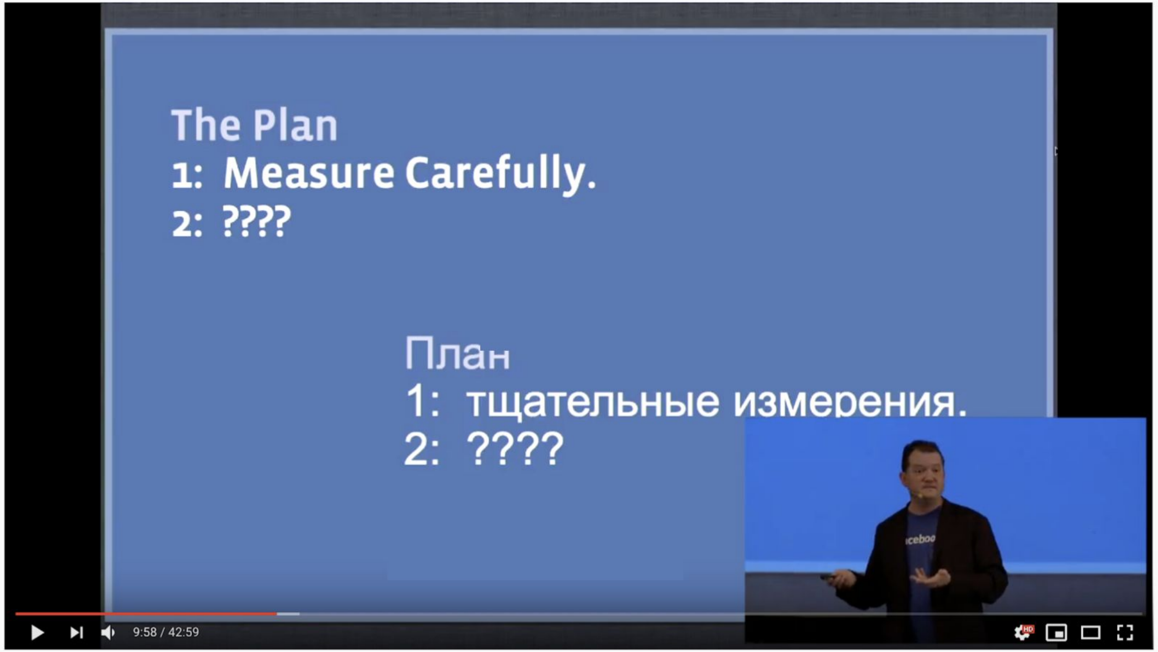
Why this matters!

How you can build something similar using (mostly) open source technologies.

What is the problem?

Larger AWS Bill
Increased scale and additional engineers doing more things.

Low visibility
Little to no insight on which internal services are spending the most money.



Mature optimization / Carlos Bueno (Facebook)

Before Infraspend...

C	E	F
Product	September 2017	October 2017
EC2	$26,000	$26,000
EC2	$25,000	$25,000
EC2	$24,000	$24,000
S3	$23,000	$23,000
EC2	$22,000	$22,000
EC2	$21,000	$21,000

The spreadsheet days

Lyft's first attempt at cloud spend visibility and management.

Shipping Infraspend 2.0

Standardized ETL pipeline

Download the Cost and Usage Report (CUR) and process/store data using same infra as rest of Lyft (Apache Airflow, Hive, Druid).

Empower future tools

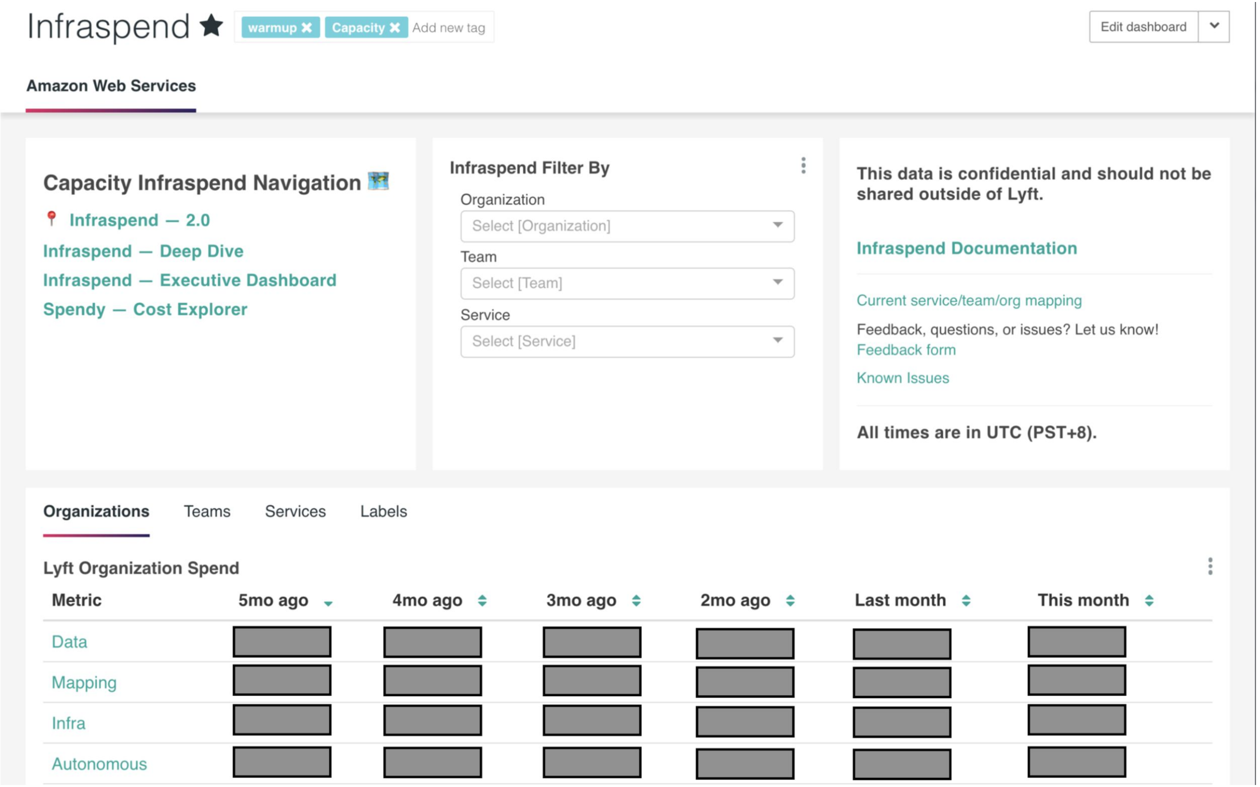
Enables ad-hoc queries, custom dashboards, and other use cases (RI analysis, capacity planning).

true_cost

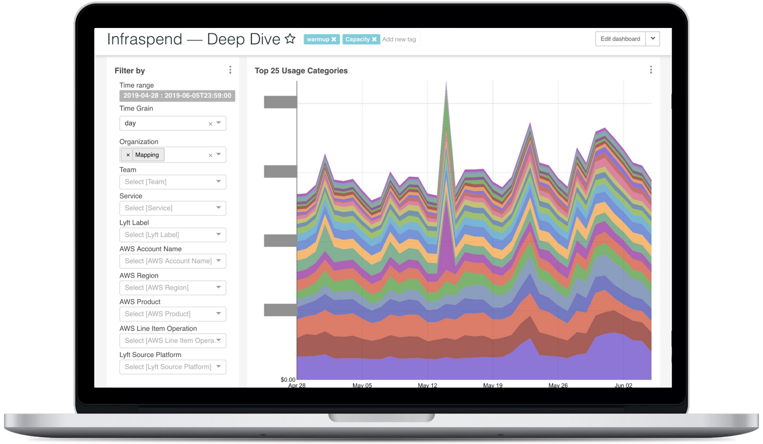
Blend together RI and EDP discounts to provide a “what you see is what you get” view of AWS spend. Allows simple and correct analysis of spend changes.

lyft_label

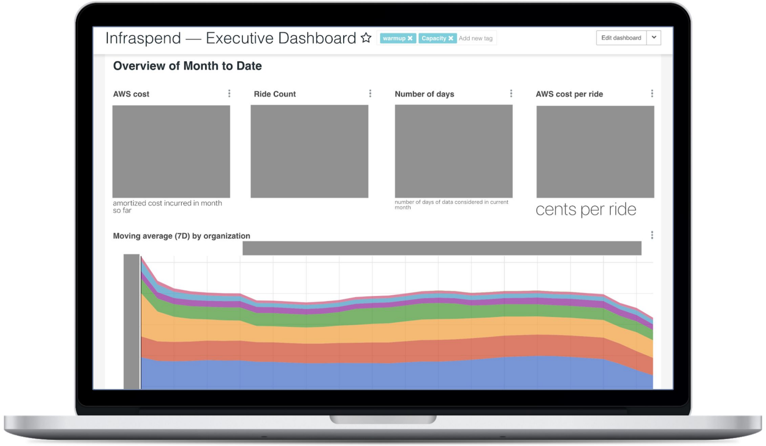
Assign usage to a Lyft-specific string based on cost allocation tags, resource IDs, and platform usage. These are then mapped to teams and orgs.


Infraspend
warmup
Capacity
Add new tag
Edit dashboard
Amazon Web Services
Capacity Infraspend Navigation
Infraspend — 2.0
Infraspend — Deep Dive
Infraspend — Executive Dashboard
Spendy — Cost Explorer
Infraspend Filter By
Organization
Select [Organization]
Team
Select [Team]
Service
Select [Service]
This data is confidential and should not be shared outside of Lyft.
Infraspend Documentation
Current service/team/org mapping
Feedback, questions, or issues? Let us know!
Feedback form
Known Issues
All times are in UTC (PST+8).
Organizations
Teams
Services
Labels
Lyft Organization Spend
Metric
5mo ago
4mo ago
3mo ago
2mo ago
Last month
This month
Data
Mapping
Infra
Autonomous

Shipping Infraspend 2.0

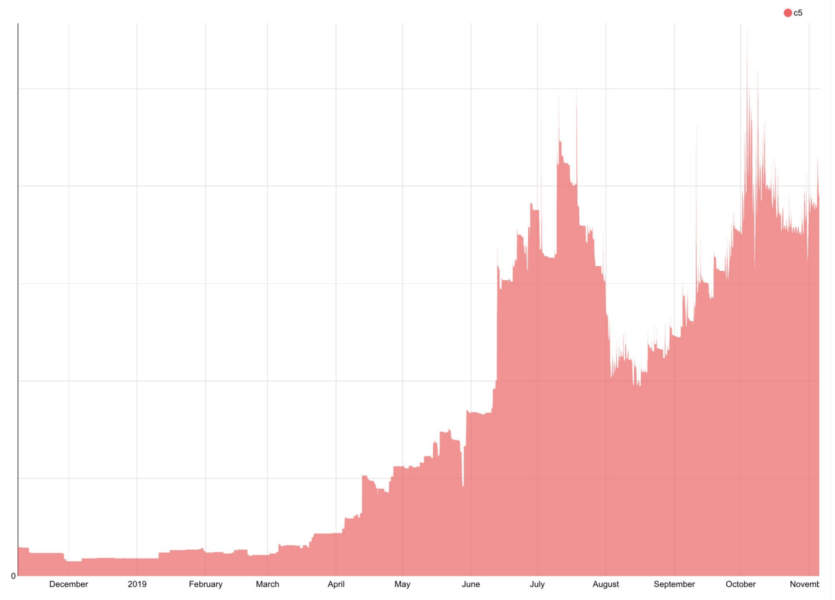


Shipping Infraspend 2.0



Kubernetes (and other platforms) reduces visibility into spend.

C5.18xlarge Usage on Core Kubernetes Clusters Over Past Year



Kubernetes @ Lyft

Dozens of clusters.

- Core Kubernetes clusters.
- Cron job clusters.
- Flyte.
- Continuous integration.
- Deploys.
- Machine Learning.
- Machine Learning for Level 5.

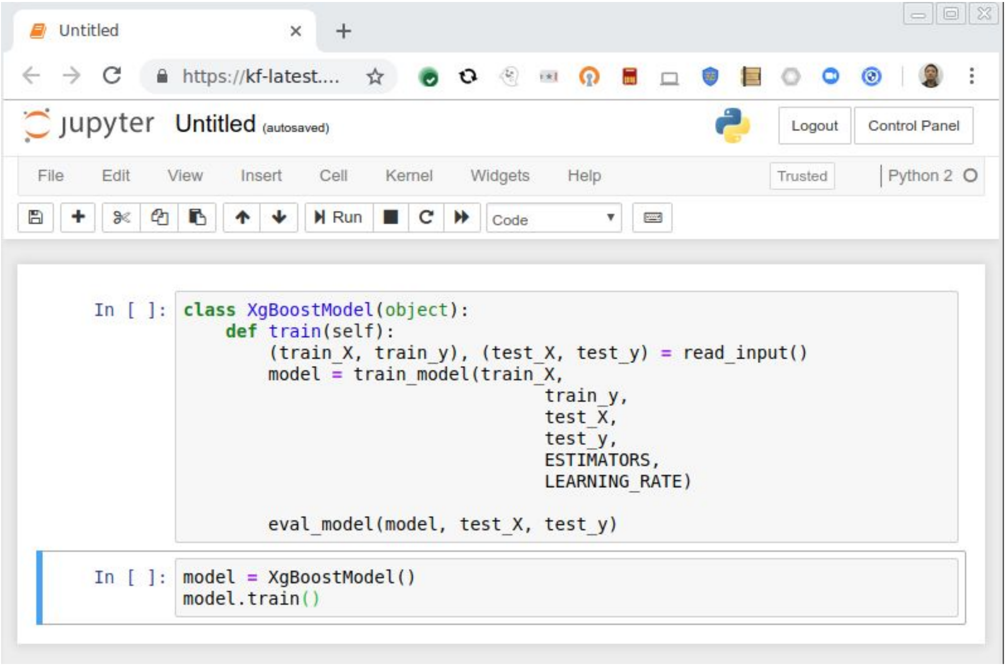
Let’s take a closer look at machine learning on Kubernetes.

A Tour of Kubeflow

Notebook Instances

Usage pattern:

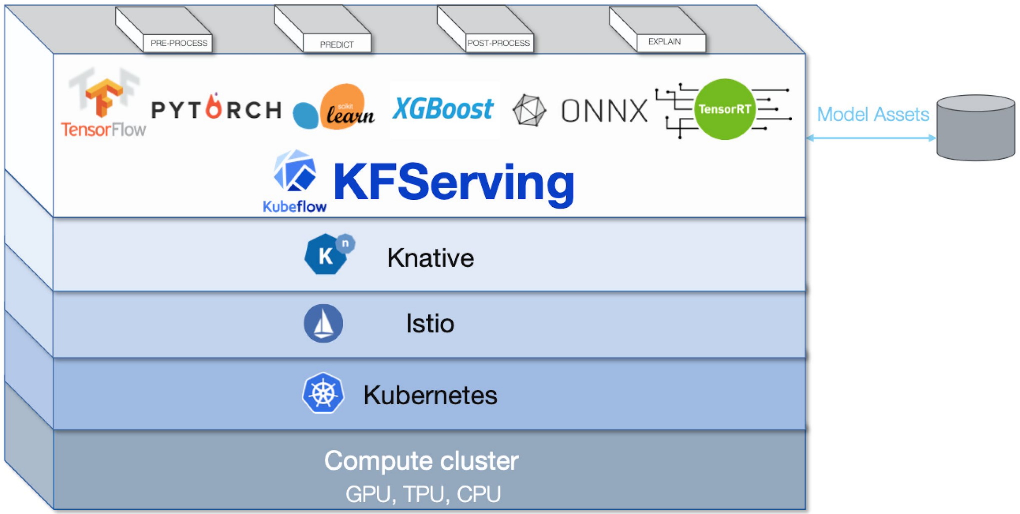
- High Availability
- Low preemption
- Multiple users
- Potentially idle notebooks



KFServing

- Scalable, Kubernetes-native intererencing

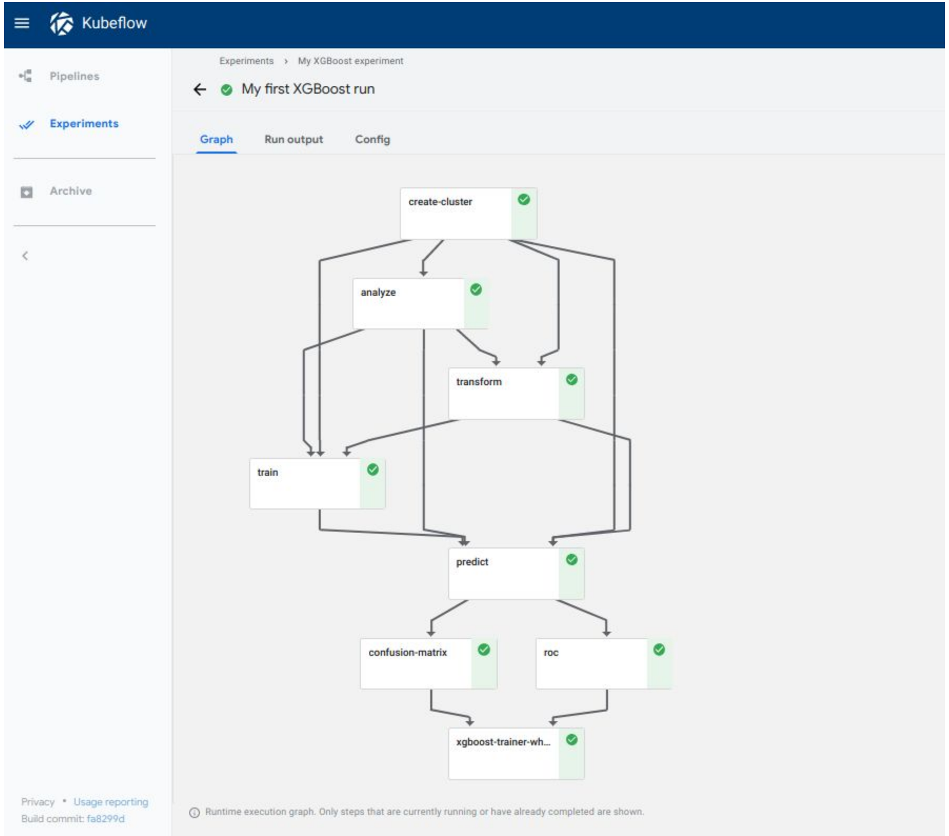
- Usage pattern:
  - High Availability
  - Quick addition of capacity
  - Potentially need GPUs



Kubeflow Pipelines

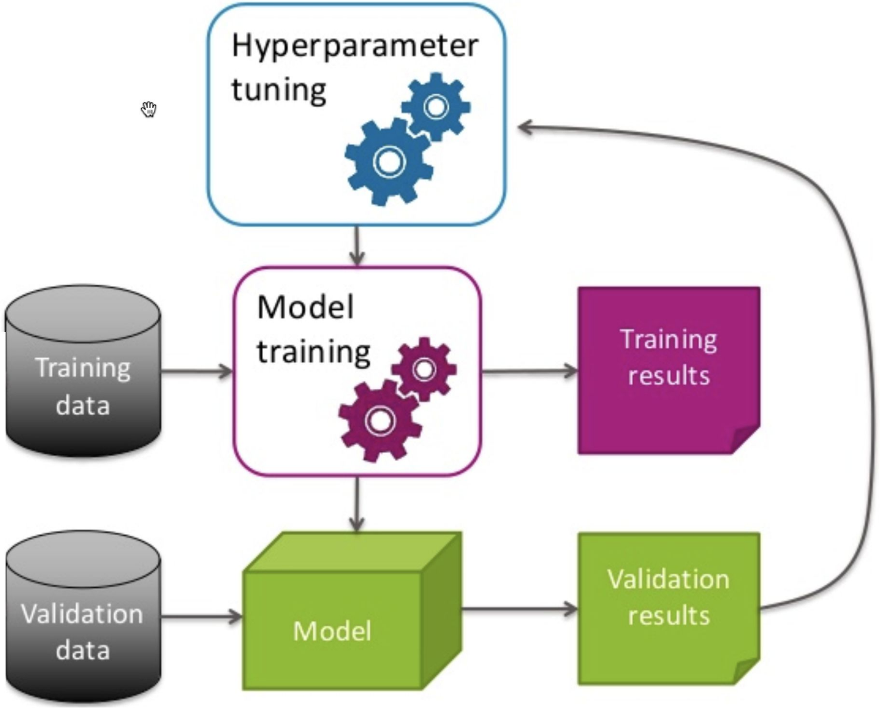
- End-to-end ML Workflows

- Usage pattern:
  - Scheduling dependencies can cause bottlenecks
  - Workflows can run regularly



Hyperparameter Optimization

- Hyperparameters are external to the model (unlike model parameters)
- Examples:
  - learning rate
  - number of layers
  - kernel type
- Hyperparameter optimization - finding the best HP values such that model performance is maximized



Source:
https://www.slideshare.net/AliceZheng3/evaluating-machine-learning-models-a-beginners-guide

How Does HP Tuning Work?

```
# Initialize search space
# Initialize model

while not objective_reached and not budget_exhausted:
    # Obtain the next set of hyperparameters
    hyperparameters = GetSuggestions()

    # Collect metrics
    metrics = RunTrial(hyperparameters)

    # Report metrics
    ReportMetrics(metrics)
```

Katib

- Framework-agnostic, production-ready hyperparameter tuning
- Usage Pattern:
  - Can be resource intensive
  - Potentially high capacity demand
  - Configurable parallelism

So What Does It All Mean?

Extending Infraspend 2.0 with Multi-Tenant Platform Attribution

What were our goals?

Modular and Extensible

Solution should extend to multiple platforms and attribution models.

Start with Kubernetes

Kubernetes usage was growing fast and visibility was necessary now.

Platform for Platforms

Provide clear documentation for how additional platforms to send us their data. Platform owners know how to attribute their platform best.

Multi-tenant Platform Concepts

Attribution Schedule

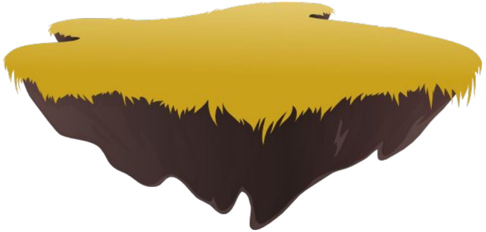
Breaks down the usage of a larger platform, per hour, by attribution label.

Practically speaking, a Hive table with certain columns.

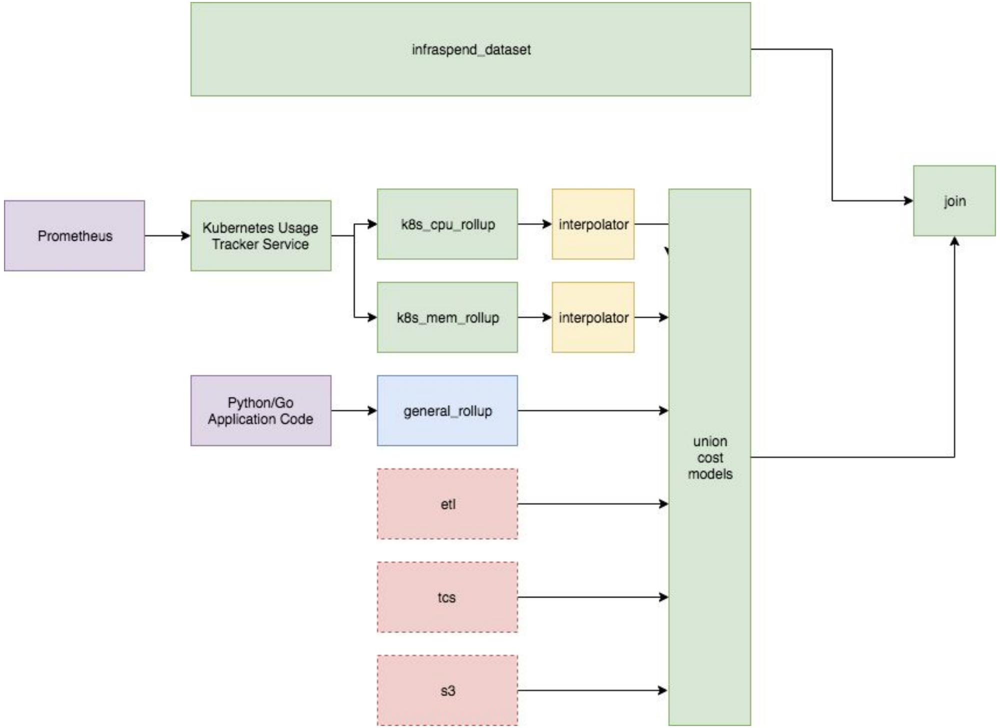
Platform Definition

Concept that ties together multiple resources under a platform and divides all resources according to the provided Allocation Schedule.

Practically speaking: a configuration file that gets ingested and used in a join.



Multi-tenant Platform Architecture



Attribution schedule generation

- Use standard Prometheus + Kubernetes pipeline that is centrally supported and maintained.
    - Either CPU or memory.

- Push custom data stream to build attribution schedule; you own the metric emission and we own the pipeline.
    - GPU, I/O, complex models.
    - Non-Kubernetes platforms.

- Provide and support own ready-made attribution schedule that has the proper format; you own everything, but you get the most control.

Attribution Schedule Properties

- Has all required expected platforms.
- Has an entry for each hour within the day.
- Total usage within a platform and sub-platform adds up to exactly 1.00.

Attribution Schedule

Hour	Cluster	Namespace	Proportion of CPU Utilized
1 PM	cluster-0	free	0.5
1 PM	cluster-0	our-first-service	0.3
1 PM	cluster-0	our-second-service	0.2
2 PM	cluster-1	free	0.1
2 PM	cluster-1	our-first-service	0.9

Zoom in on providing namespace level attribution for Kubernetes.

Infraspend Data

Hour	lyft_label	Product	Cost
1 PM	cluster-0	EC2	$1.00
2 PM	cluster-1	EC2	$2.00
2 PM	some-other-cluster	EC2	$1.00

Enriched Infraspend Data

Hour	lyft_label	Product	Cost
1 PM	free (k8s: cluster-0)	EC2	$0.50
1 PM	our-first-service (k8s: cluster-0)	EC2	$0.30
1 PM	our-second-service (k8s: cluster-1)	EC2	$0.20
2 PM	free (k8s: cluster-1)	EC2	$0.20
2 PM	our-first-service (k8s: cluster-1)	EC2	$1.80

Attribution Schedule

Hour	Cluster	Namespace	Proportion of CPU Utilized
1 PM	cluster-0	free	0.5
1 PM	cluster-0	our-first-service	0.3
1 PM	cluster-0	our-second-service	0.2
2 PM	cluster-1	free	0.1
2 PM	cluster-1	our-first-service	0.9

Cost Models

- **CPU Allocation.**

- Memory Allocation.

- GPU Allocation.

$$\frac{\textit{number of CPU cores requested by the namespace by all running pods on the cluster}}{\textit{number of CPU cores available on the cluster across all active instances}}$$

- max(CPU, memory, GPU).

- Deconstruct from cloud service provider and weigh all resource costs.

- I/O, storage, etc.

Mind the Unallocated Capacity

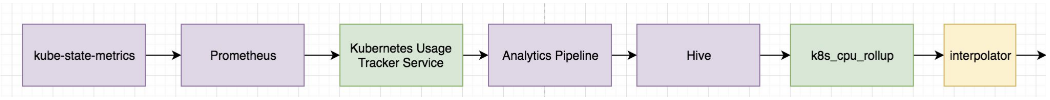
- **CPU Allocation.**

$$\begin{array}{r} \text{number of } CPU \text{ cores available on the cluster across all active instances} \\ -\quad \text{number of } CPU \text{ cores allocated to running pods} \\ \hline \text{unallocated cluster } CPU \text{ capacity} \end{array}$$

Infraspend is about allocation, not efficiency of that allocation.

We built additional infrastructure and products to monitor efficiency.

Collecting Kubernetes Metrics



Kubernetes Usage Tracker Service

Light-weight service, scrapes metrics from Prometheus about cluster capacity, pod labels, node labels, and memory and CPU utilization.

Example Queries

- kube_pod_container_resource_requests_cpu_cores * on(pod) group_left kube_pod_status_phase{phase="Running", job="kubernetes-service-endpoints"}
- kube_node_status_capacity

Lessons

- Filter metrics for only running pods.
- Include instance type as dimension. Attribute pods correctly to instance.
- Have adequate monitoring by metric, cluster, region, etc.

Computing Allocation Schedule

Rollup Prometheus Metrics in Hive

Light-weight service, scrapes metrics from Prometheus about cluster capacity, pod labels, node labels, and memory and CPU utilization.

Interpolate and Validate in Python

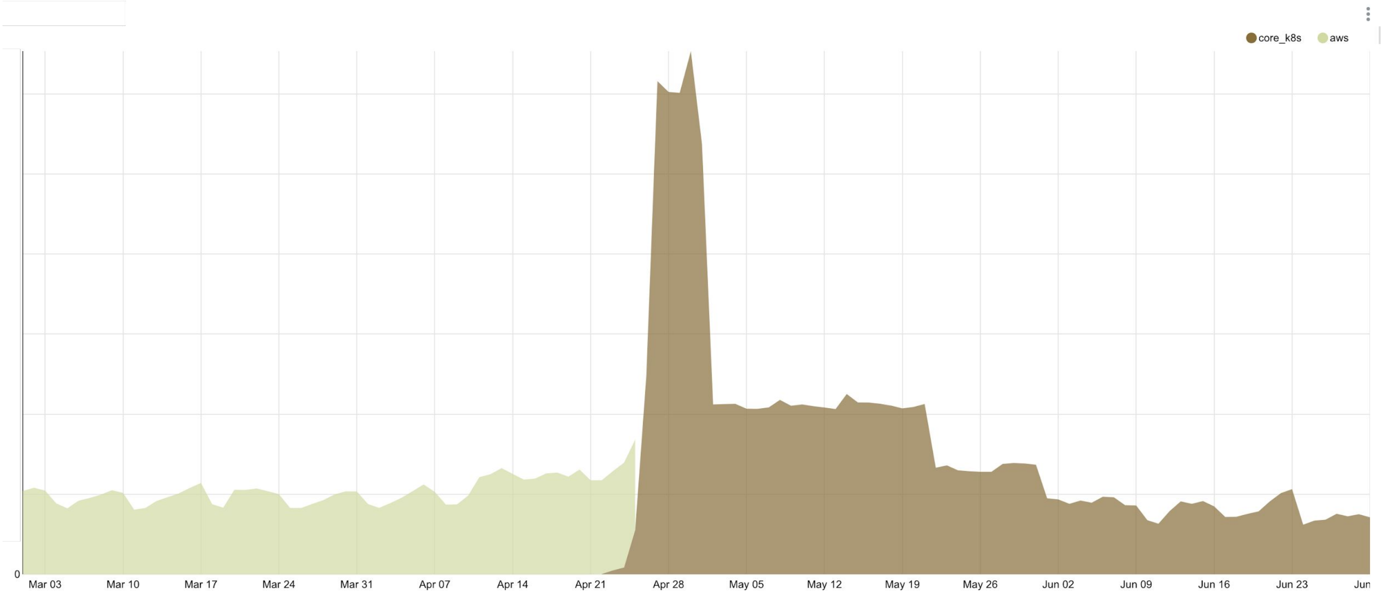
Light-weight service, scrapes metrics from Prometheus about cluster capacity, pod labels, node labels, and memory and CPU utilization.

Challenges ➡️ Lessons

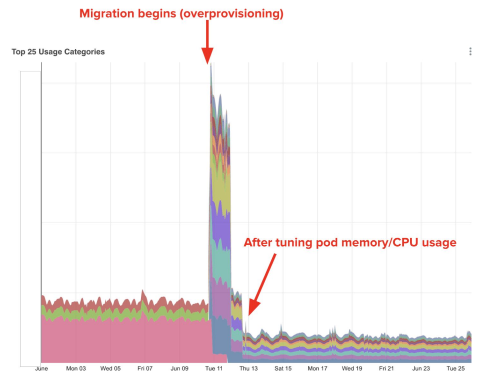
Challenge	Lesson
Lots of components; at scale, each one misbehaves sometimes.	● Understand dependencies. ● No substitute for building the system. ● Assume every step is broken and do sanity checks at each one.
Operational load is high across lots of platforms.	● Log rate of dataflow. ● Interpolate data so that small blips don’t break Infraspend. ● Build automated notifications to platform owners when their systems are not functioning properly.

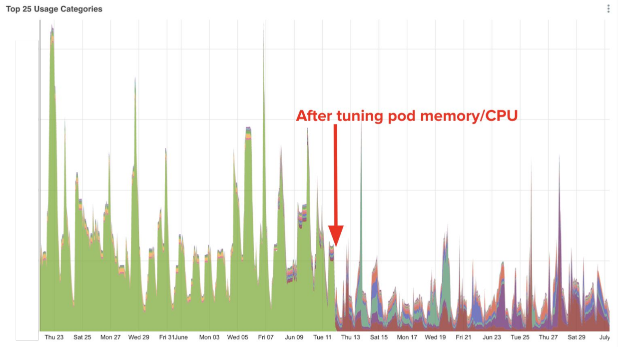
Infraspend 2.0 with Kubernetes

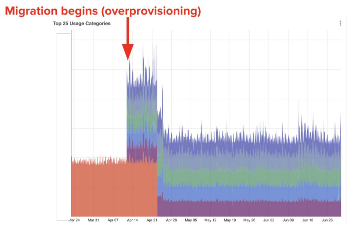
Migrating to Kubernetes

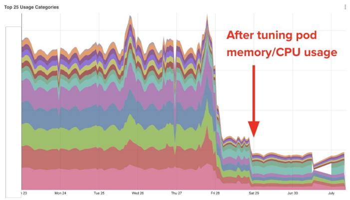


Tracking Migration Impact

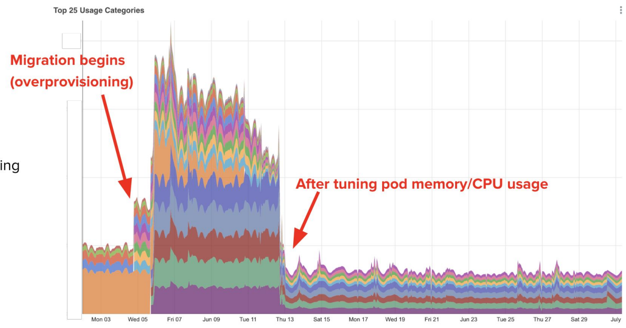


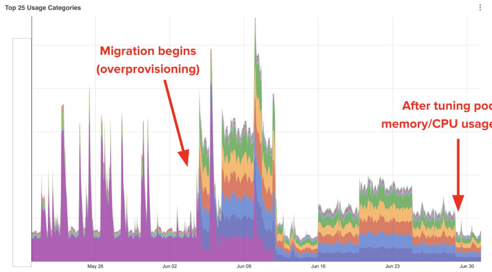


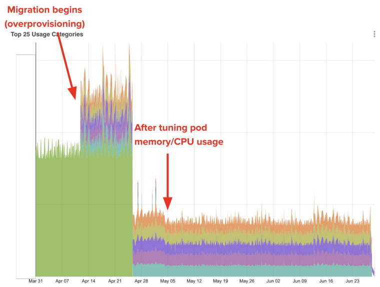




ing







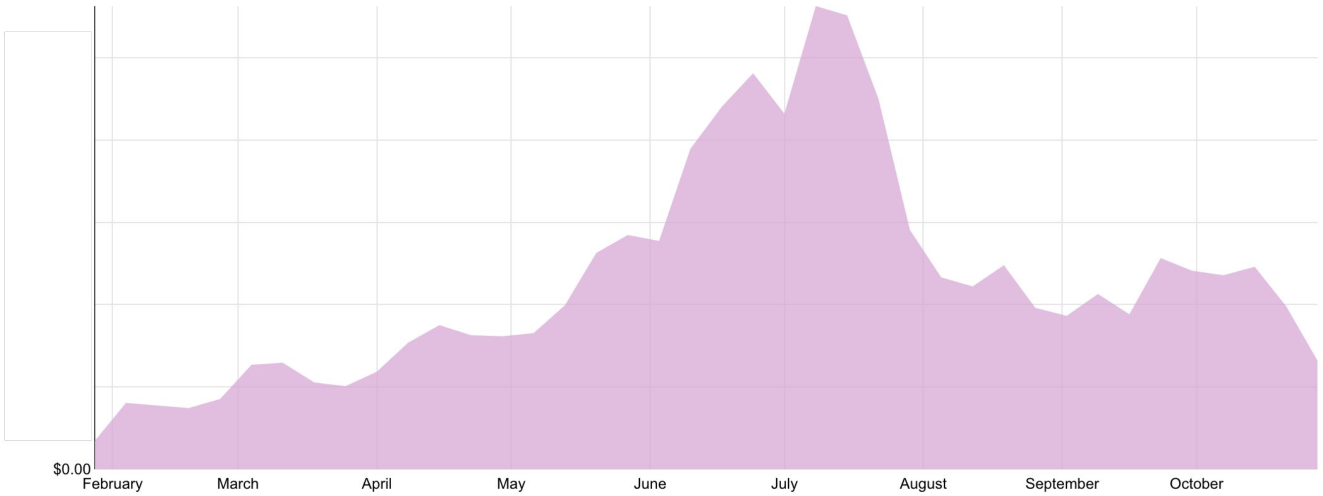
Impact: side-by-side visualization of Kubernetes costs.

Impact: allow engineering teams to track their costs across cloud products and platforms.

Lesson: for migrations, enforce namespaces match the service name to naturally tie usage together.

Lesson: set minimum number of pods per cluster gradually lower to ensure that the system is still reliable.

Unallocated Cluster Capacity



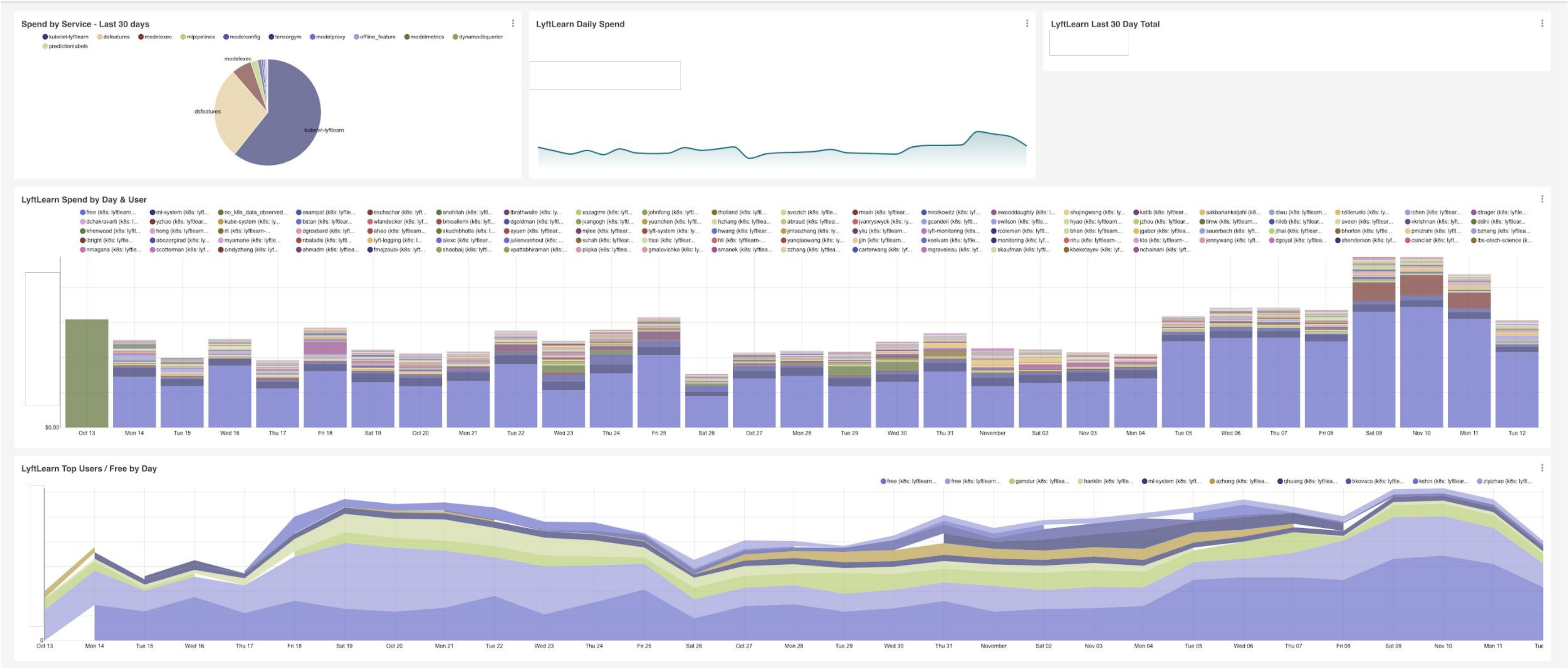
Impact: raise awareness of unallocated capacity across platforms and enable tracking.

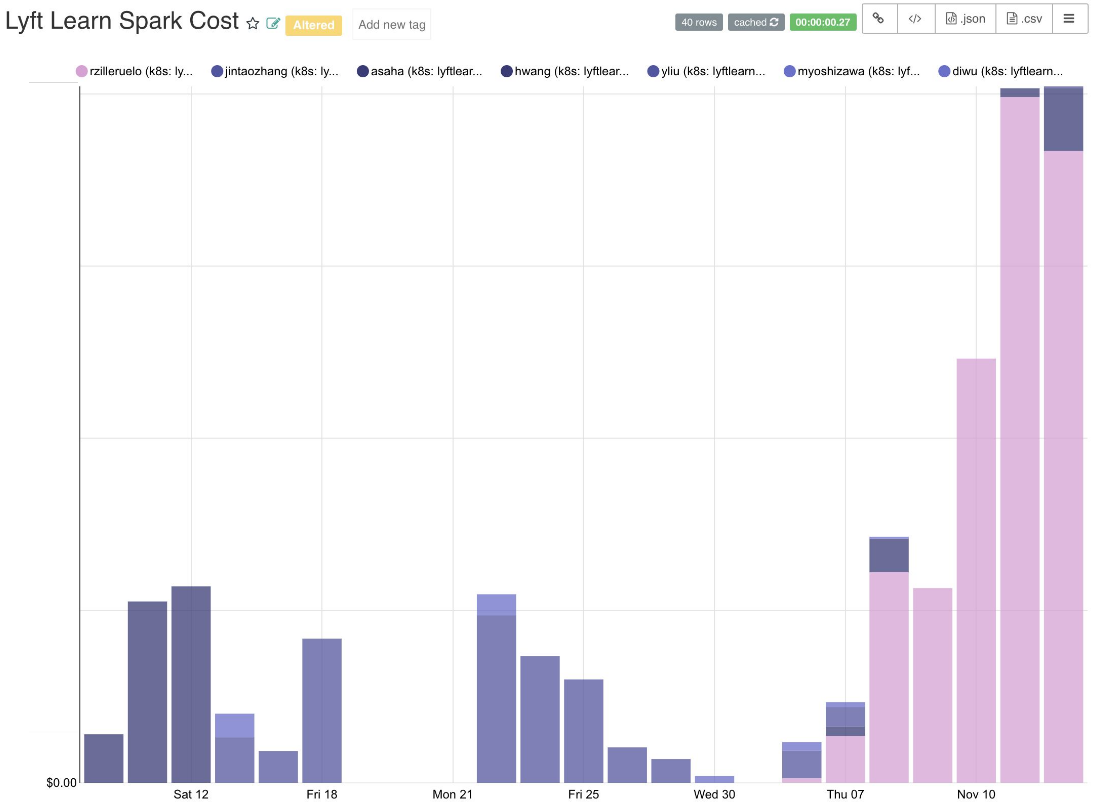
Lessons to lower free space:

- Tune cluster scaling policies.
- Tune pod scaling policies.
- Choose more suitable scheduler.
- Deploy more services.

Expanding Kubernetes Allocation Tracking

- Support more allocation schedules for Kubernetes.
- Container name as dimension in Kubernetes data.
- Custom pod labels as dimension in Kubernetes data.
- Work with teams to help them build custom views into the data.


Spend by Service - Last 30 days
kubelet-lyftlearn
dsfeatures
modelexec
LyftLearn Daily Spend
LyftLearn Last 30 Day Total
LyftLearn Spend by Day & User
LyftLearn Top Users / Free by Day



Lyft Learn Spark Cost
Altered
Add new tag
40 rows
cached
00:00:00.27
.json
.csv
rzilleruelo (k8s: ly...
jintaozhang (k8s: ly...
asaha (k8s: lyftlear...
hwang (k8s: lyftlear...
yliu (k8s: lyftlearn...
myoshizawa (k8s: lyf...
diwu (k8s: lyftlearn...
$0.00
Sat 12
Fri 18
Mon 21
Fri 25
Wed 30
Thu 07
Nov 10

Mostly Open Source

kubernetes/kube-state-metrics: Add-on agent to ... - GitHub
https://github.com › kubernetes › kube-state-metrics ▾
kube-state-metrics is a simple service that listens to the **Kubernetes** API server and generates **metrics** about the **state** of the objects. ... It is not focused on the health of the individual **Kubernetes** components, but rather on the health of the various objects inside, such as deployments, nodes and pods.

Docs
Pod Metrics - Node Metrics - Deployment Metrics - **...**

33 releases
... generate and expose cluster-level metrics. - kubernetes/kube ...

README.md
... to generate and expose cluster-level metrics. - kubernetes/kube ...

kube-state-metrics ...
... to generate and expose cluster-level metrics. - kubernetes/kube ...

Pod Metrics
... to generate and expose cluster-level metrics. - kubernetes/kube ...

Node Metrics
... to generate and expose cluster-level metrics. - kubernetes/kube ...

More results from github.com »

HIVE

druid

Superset

So you can replicate it at your company!

Looking ahead!

- More platforms.
- Finer granularity.
- Deeper insights beyond allocation.
- As close as possible to real-time.
- Deeper integration with frameworks, such as experimentation, so we can track the cost of features across multiple services.

Thank you!

Questions?

Join us for some local beer, wine, and tacos!

Lyft Happy Hour

Date: Tuesday, Nov 19
Time: 7pm-10pm
Where: Thorn Barrio Logan (1745 National Avenue, San Diego, CA 92113)

RSVP: https://lyft-kubecon.splashthat.com/ (you can also register at the door)

Convert Model to Lyftlearn Template

Hyperparameters

```
class Model(object):
  HYPERPARAMETERS = [{'name':'dropout','type':'float', 'default_value':0.2},
                     {'name':'layers', 'type':'int', 'default_value':3}]

  def __init__(self, hyperparameters = None):
    hyperparameters = hyperparameters or {}
    # Read and convert hyperparameters
    self.dropout = hyperparameters["dropout"]
    self.layers = hyperparameters["layers"]

  def train(self):
    pass

  def init_predict():
    pass

  def predict(self, request_data):
    pass

  def batch_predict(self):
    pass
```

Convert Model to Lyftlearn Template

Train Function

```
class Model(object):
  HYPERPARAMETERS = [{'name':'dropout','type':'float', 'default_value':0.2},
                     {'name':'layers', 'type':'int', 'default_value':3}]

  def __init__(self, hyperparameters = None):
    hyperparameters = hyperparameters or {}
    # Read and convert hyperparameters
    self.dropout = hyperparameters["dropout"]
    self.layers = hyperparameters["layers"]

  def train(self):
    pass

  def init_predict():
    pass

  def predict(self, request_data):
    pass

  def batch_predict(self):
    pass
```

Convert Model to Lyftlearn Template

Predict functions

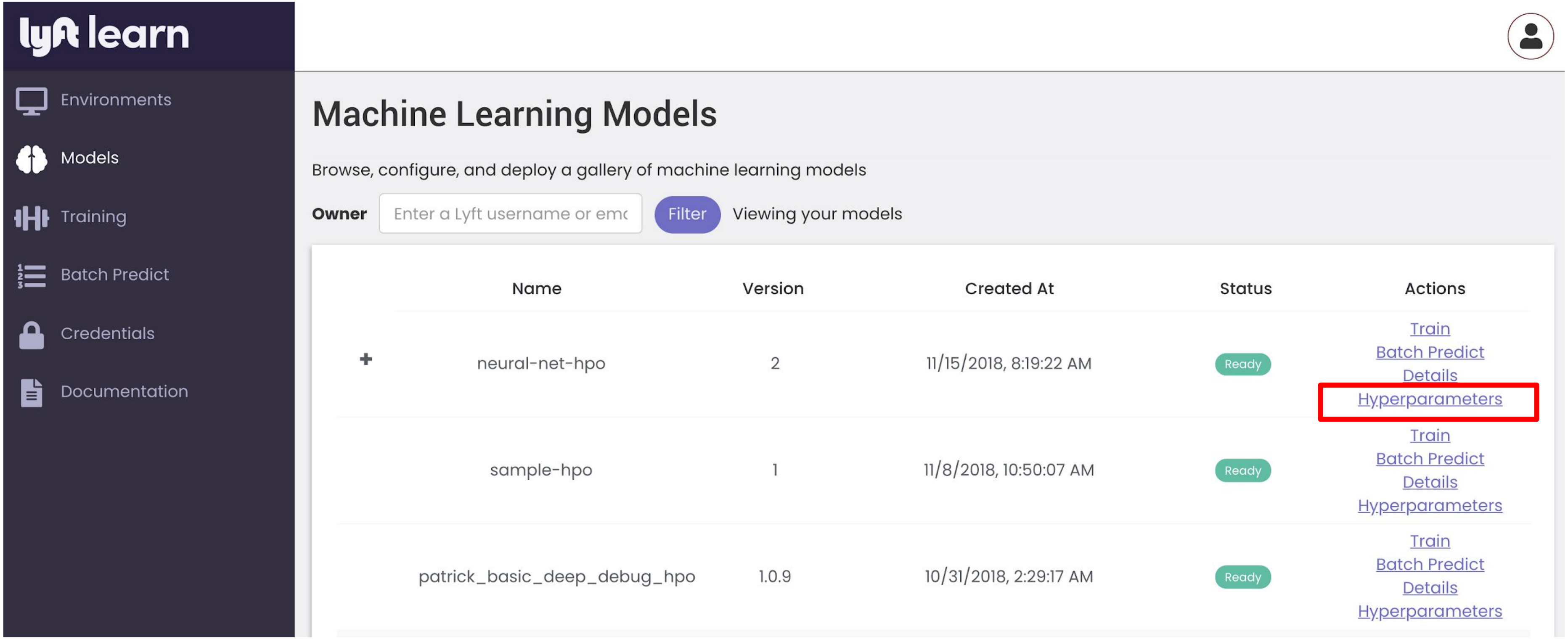
```
class Model(object):
  HYPERPARAMETERS = [{'name':'dropout','type':'float', 'default_value':0.2},
                     {'name':'layers', 'type':'int', 'default_value':3}]

  def __init__(self, hyperparameters = None):
    hyperparameters = hyperparameters or {}
    # Read and convert hyperparameters
    self.dropout = hyperparameters["dropout"]
    self.layers = hyperparameters["layers"]

  def train(self):
    pass

  def init_predict():
    pass

  def predict(self, request_data):
    pass

  def batch_predict(self):
    pass
```

Train Function example

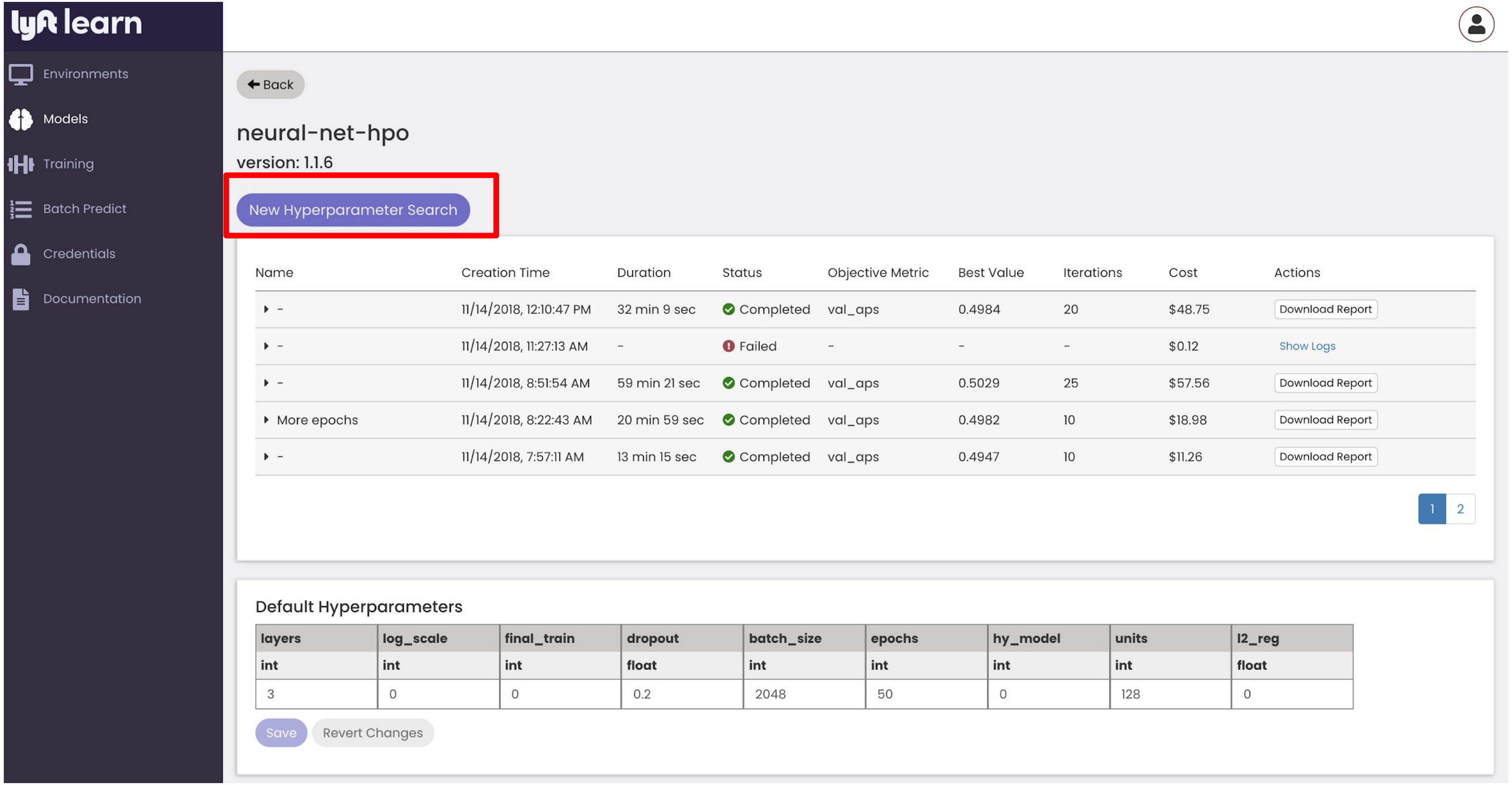
Metrics
(key for HPO)

```
from lyftlearnclient.metrics import Metrics

def train(self):
        df = presto.DatabaseTool().query('select foo from bar')
        labels = df['duration']
        training_data = df.drop(columns=['duration'])
        x_train, x_validate, y_train, y_validate = model_selection.train_test_split(
            training_data, labels, test_size=0.1)
        rf = RandomForestRegressor(n_estimators=self.n_estimators,
             max_features=self.max_features)
        rf.fit(x_train, y_train)
        self.model = rf
        train_mse = sklearn.metrics.mean_squared_error(y_train, rf.predict(x_train))
        validate_mse = sklearn.metrics.mean_squared_error(y_validate,
             rf.predict(x_validate))

        metrics.emit('train_rms', train_rms)
        metrics.emit('validate_rms', validate_rms)
        try:
            with s3.open(MODEL_PATH, mode='wb') as f:
                joblib.dump(rf, f)
        except Exception as e:
            print('Failed to save model', e)
```

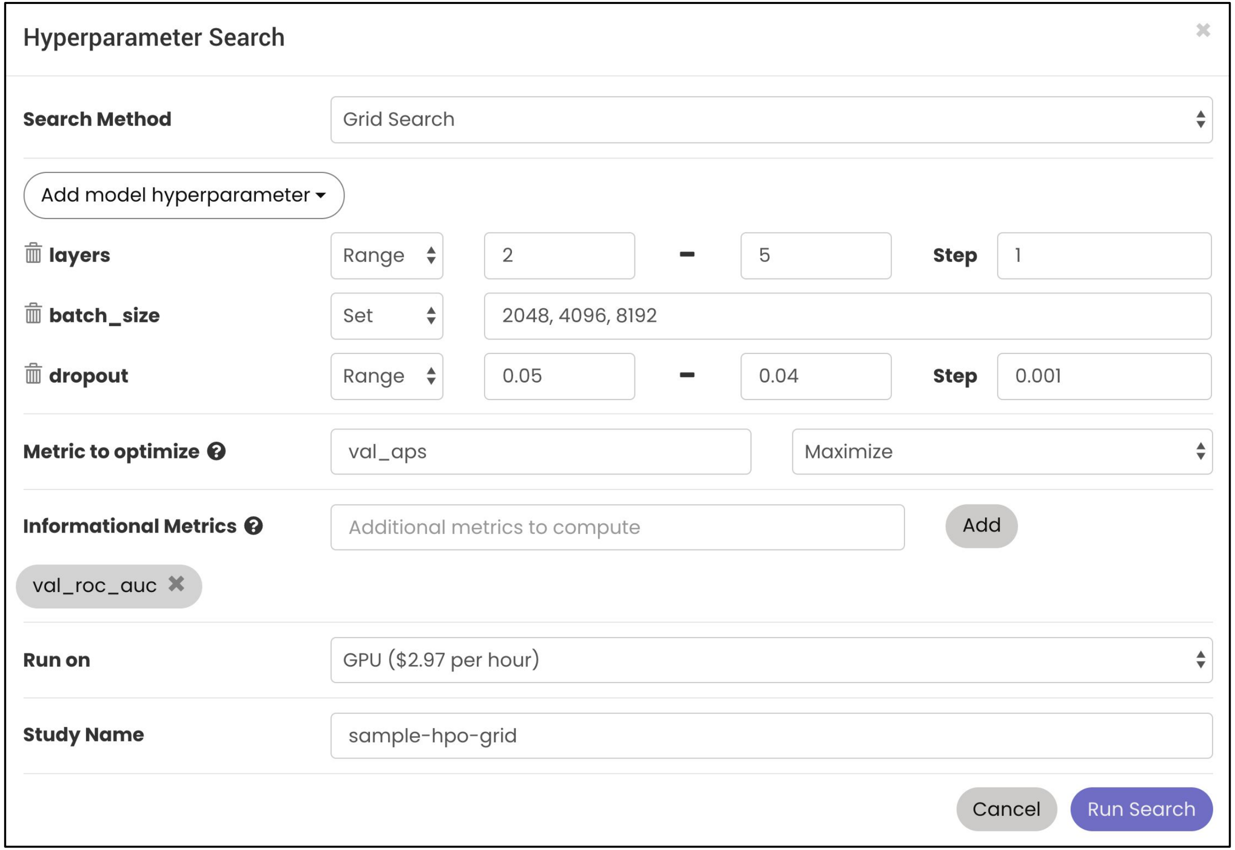
Automatic Hyperparameter tuning



Automatic Hyperparameter tuning

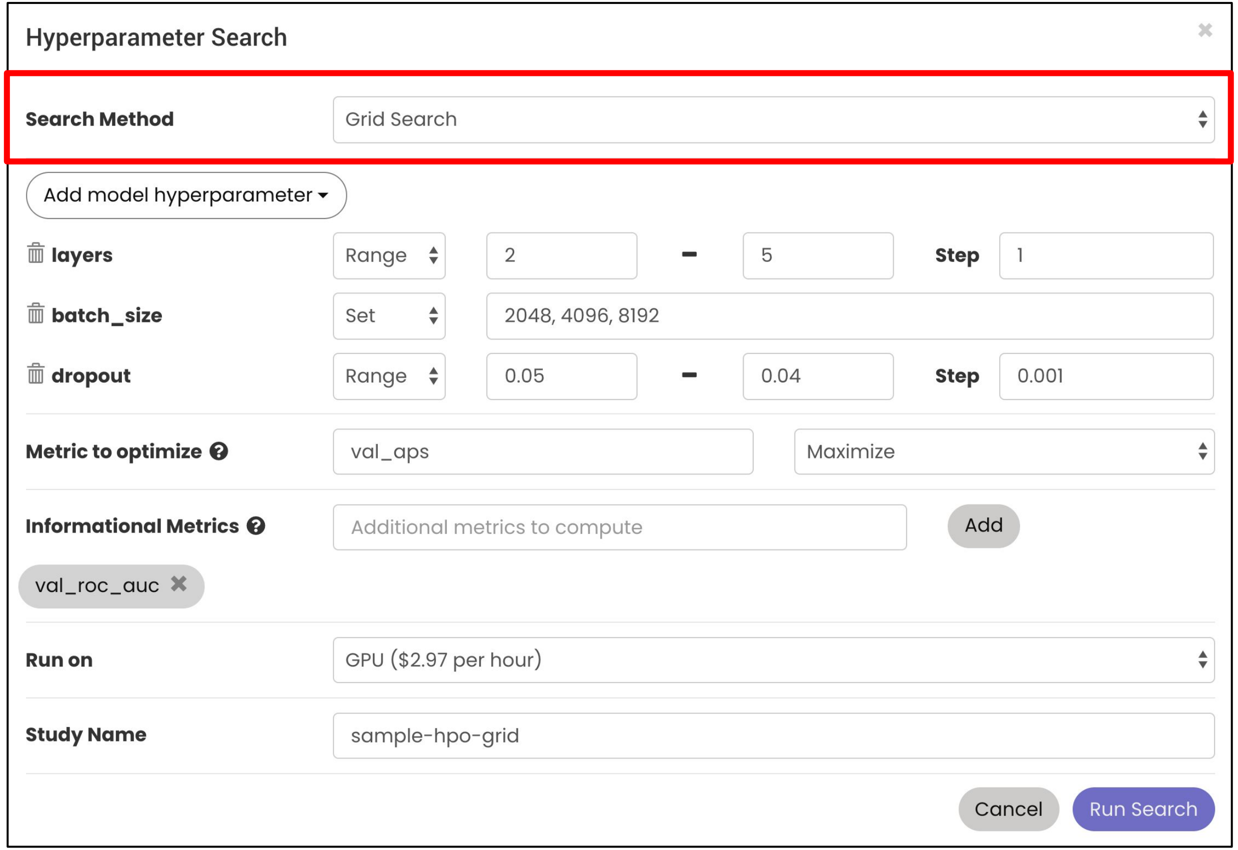


Automatic Hyperparameter tuning



Automatic Hyperparameter tuning

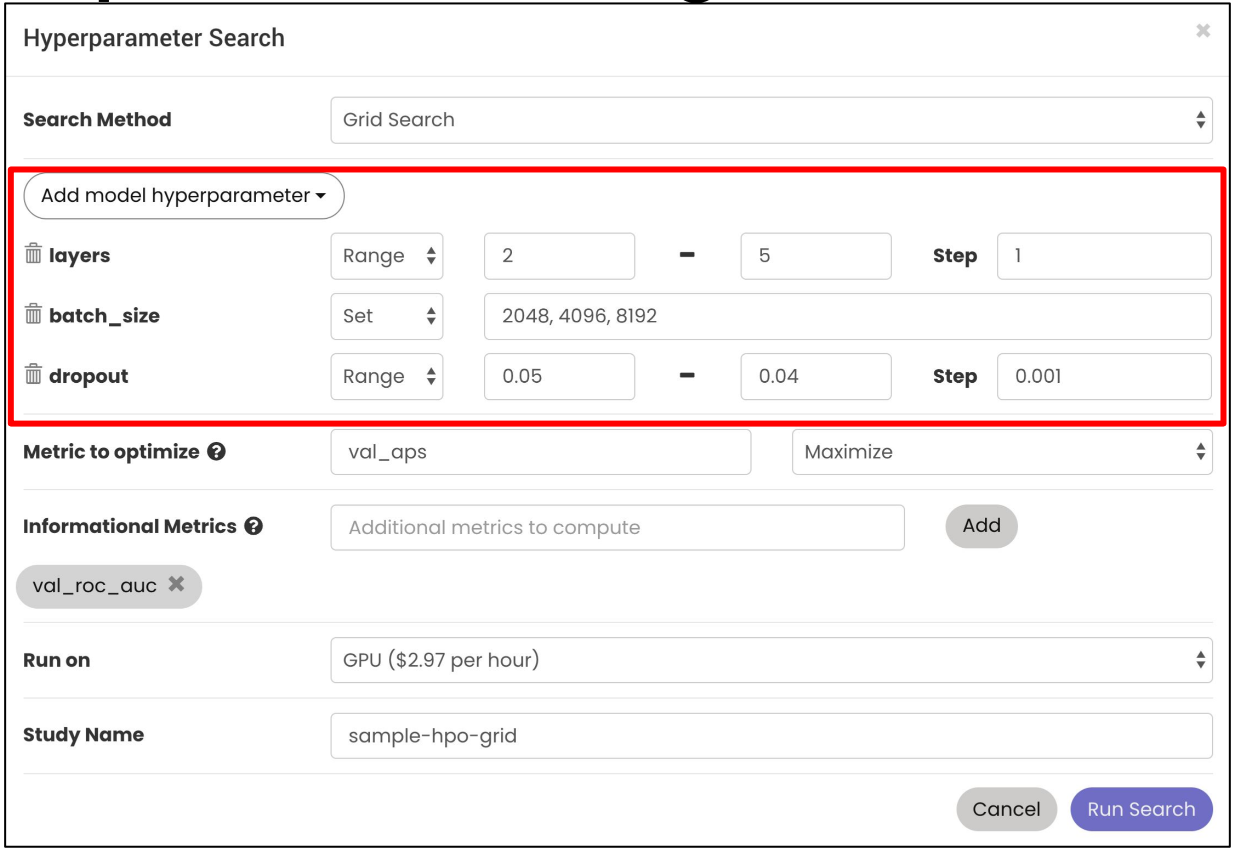
Select the Search Algorithm



Automatic Hyperparameter tuning

Select the hyperparameters to be tuned

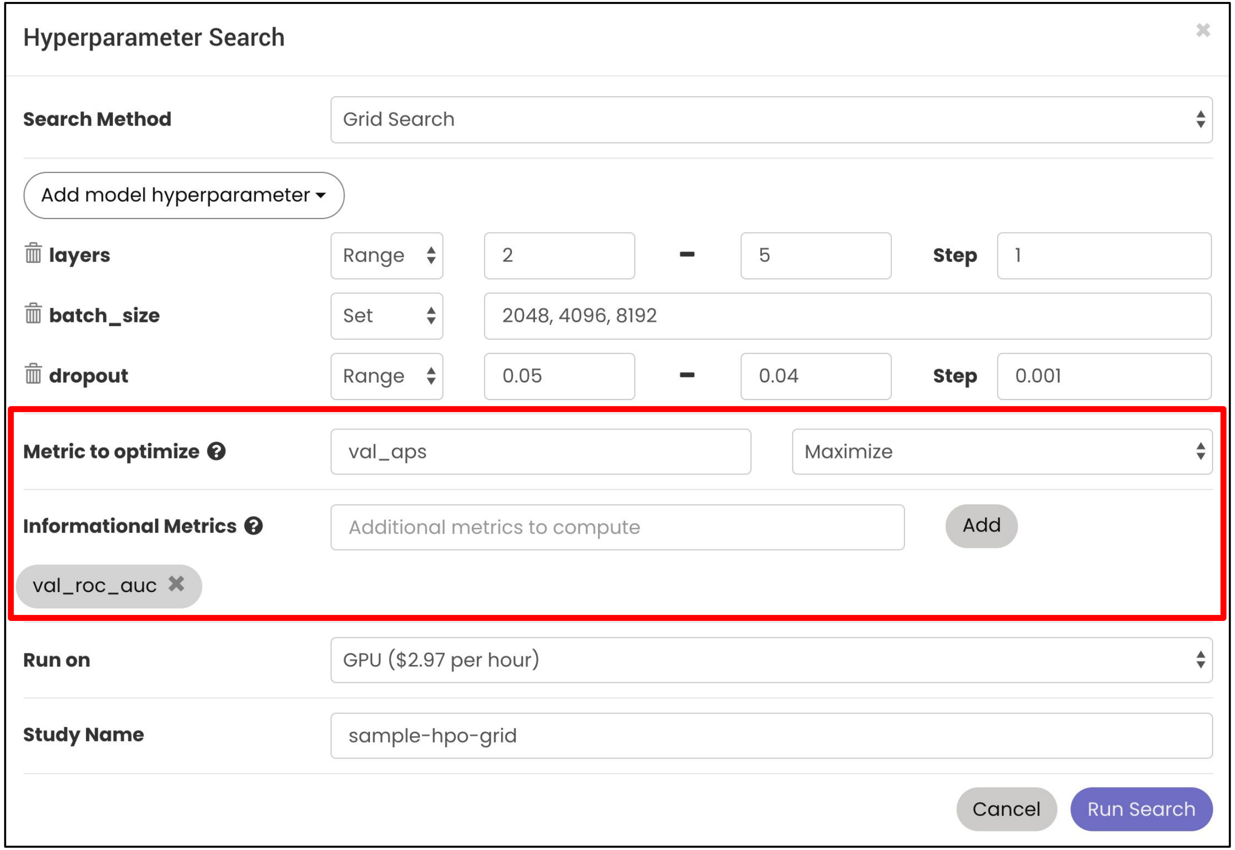
Define the search space for each



Automatic Hyperparameter tuning

Define the primary metric to be used for optimization

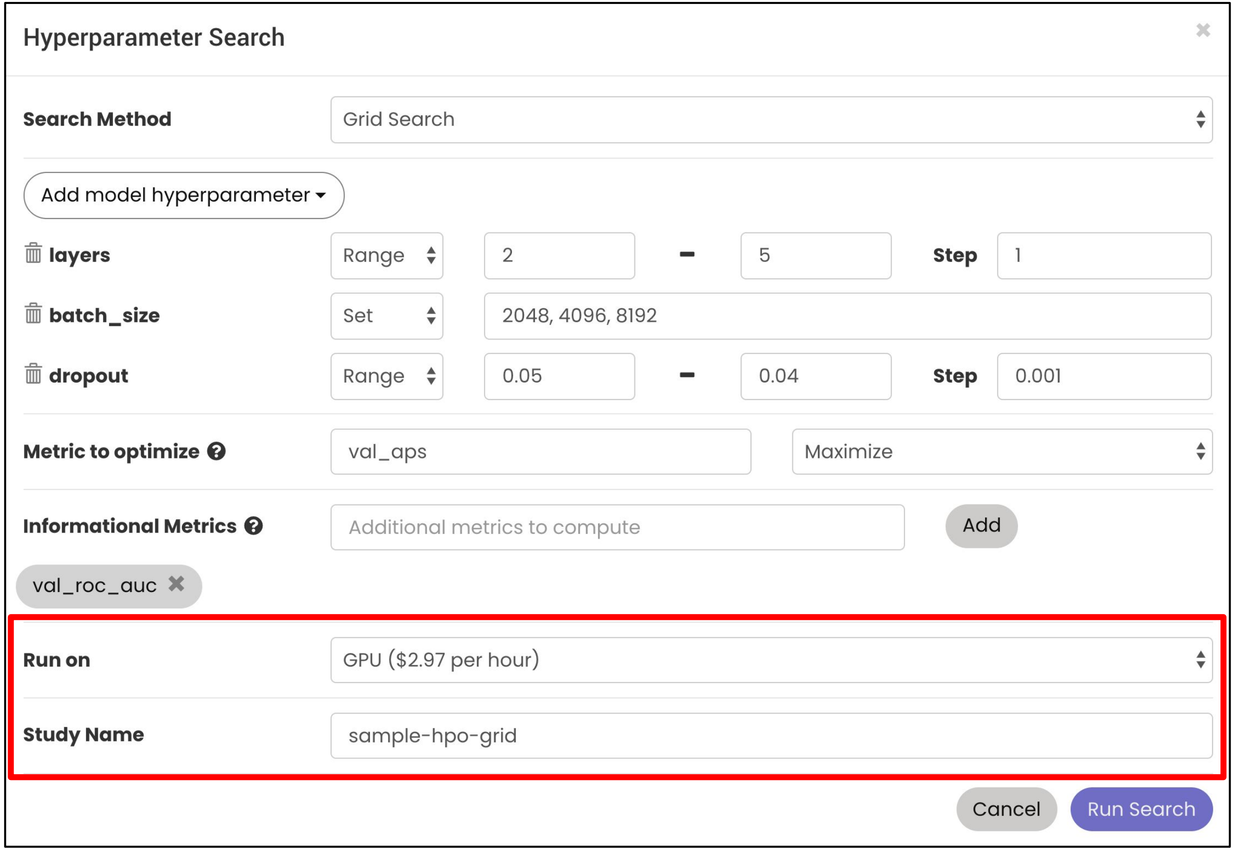
Declare additional metrics you would like to be tracked.



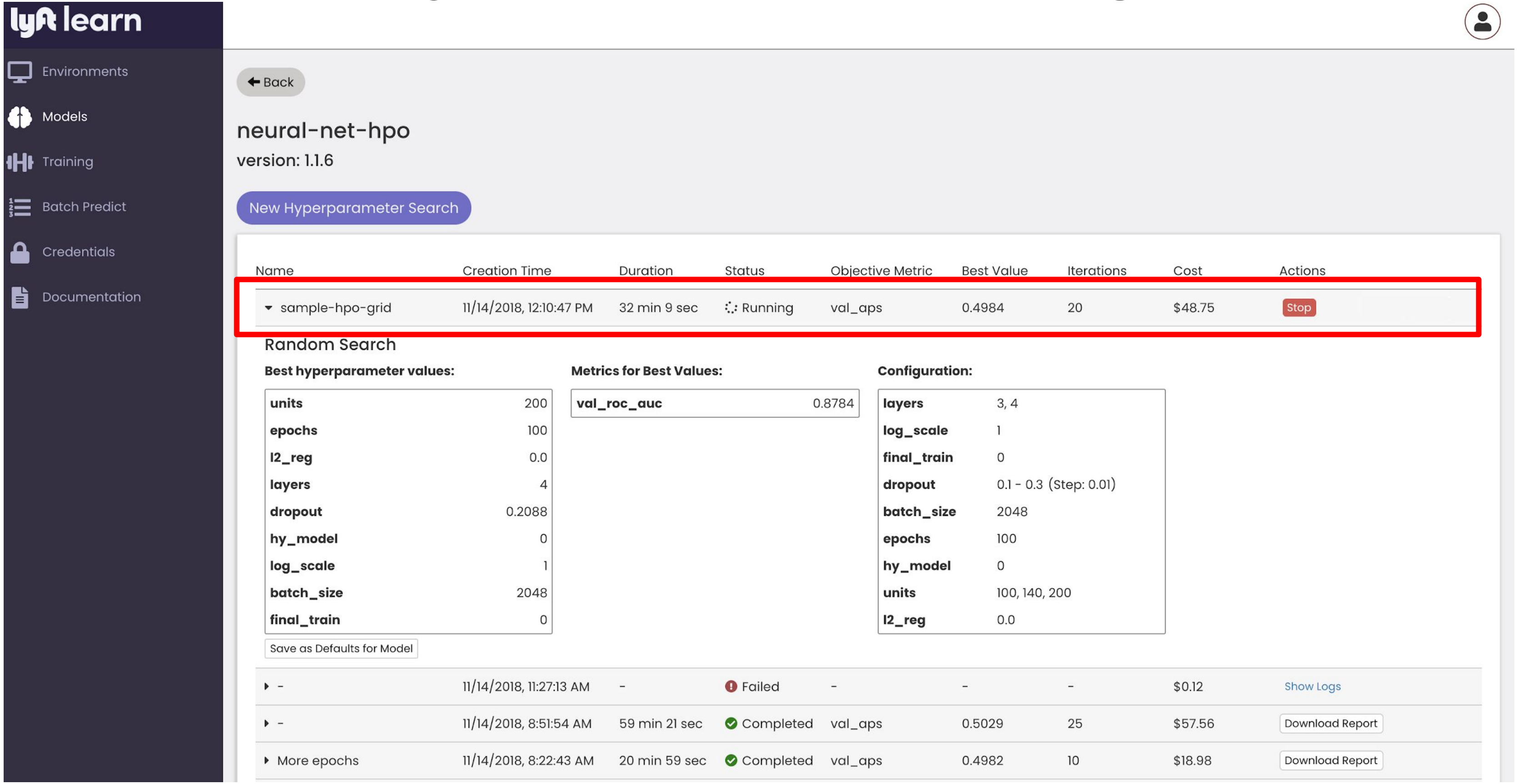
Automatic Hyperparameter tuning

Specify the Compute resources

Add an (optional) name



Automatic Hyperparameter tuning



Automatic Hyperparameter tuning