Scaling model training

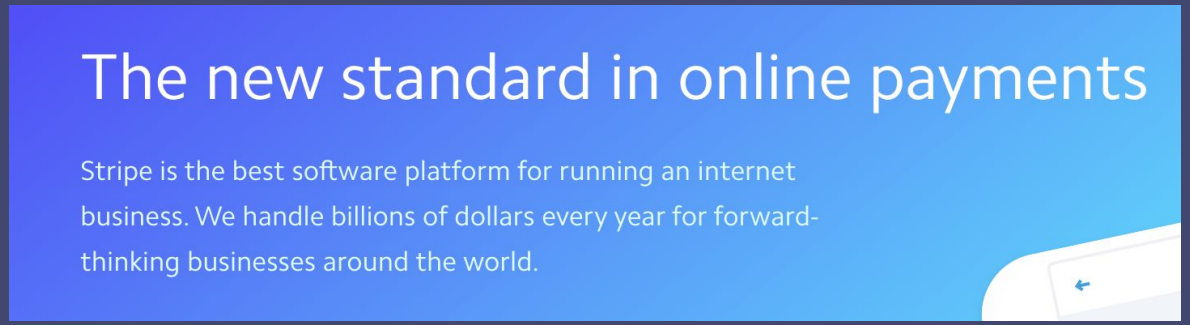
From flexible training APIs to resource management with Kubernetes

Kelley Rivoire, Stripe

stripe

Real World Machine Learning (@ Stripe)

- Stripe provides a toolkit to start and run an internet business

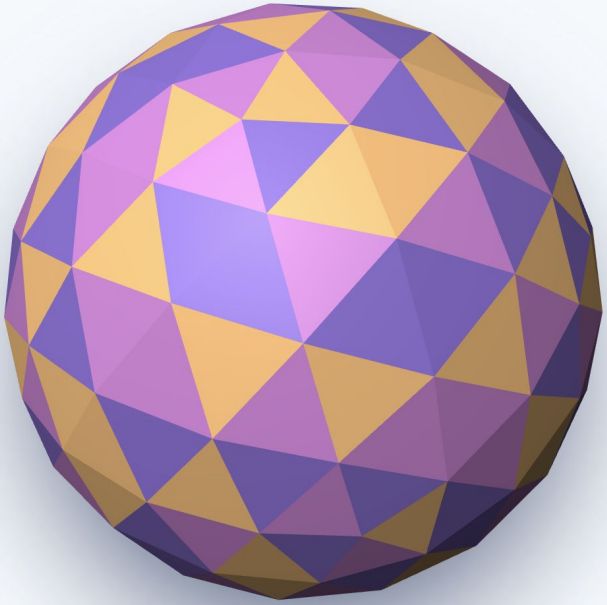


- We need to make decisions *quickly* and *at scale*
- Our actions affect *real businesses*

stripe

stripe

RADAR



Trained with hundreds of billions of data points

Stripe processes payments from 195 countries for every industry, company size, and business model. Even if a card is new to your business, there's an 89% chance it's been seen before on the Stripe network.

Strength in numbers

By learning from millions of global businesses processing billions in payments each year, Radar can assign risk scores to every payment and automatically block many high-risk payments.

Better ML outcomes with Stripe-scale data

Radar scans every payment using thousands of signals from across the Stripe network to help detect and prevent fraud—even before it hits your business.

stripe

stripe

BILLING

REDUCE DECLINED PAYMENTS BY UP TO 45%

Nearly a quarter of churn is caused by missed payments or declined cards. In 2017, Stripe's recovery tools reduced payment declines for users by 45% on average and increased revenue by 10% on average.

Automatic card updater

Stripe works directly with card networks to update payment details with new card numbers or expiry dates.

Smart retry logic

Stripe uses machine learning algorithms that train on data from across the Stripe network to optimize retry logic and minimize failed payments.

Payment reminders and overdue notices

Maximize your chances of getting paid with pre-built email reminders for missed or overdue payments.

stripe

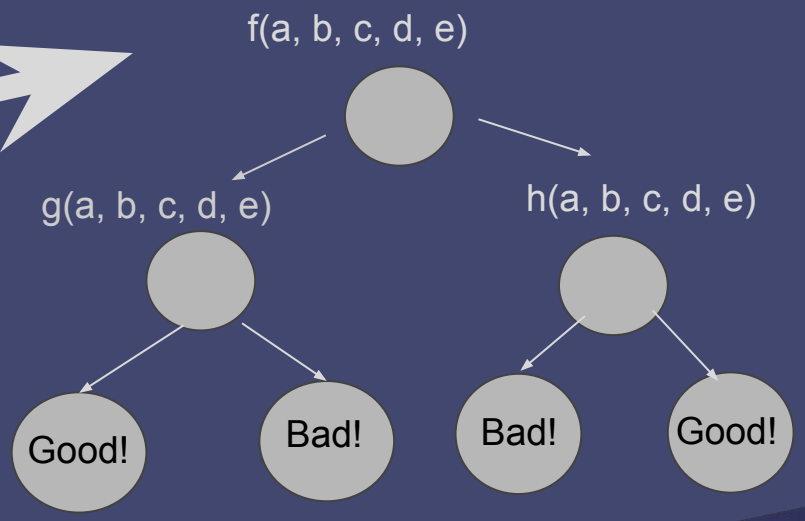
Model training

stripe

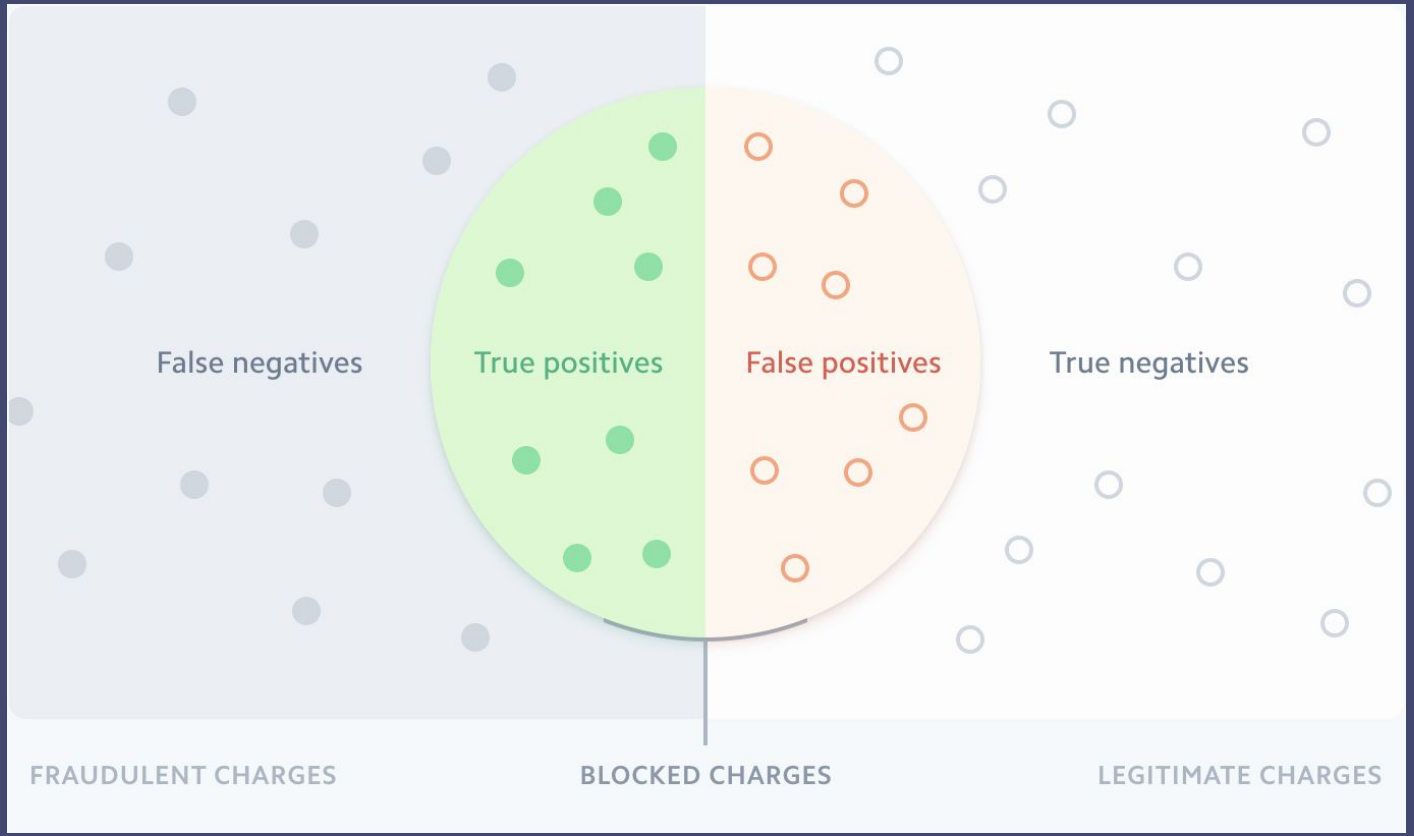
Amount in USD	Card country	Countries card used from (24h)	Fraud?
$10.00	US	1	No
$10.00	CA	2	No
$10.00	CA	1	No
$10.00	US	1	Yes
$30.00	US	1	Yes
$99.00	CA	1	Yes
$15.00	CA	3	Yes
$70.00	US	1	No

Toy model of ML

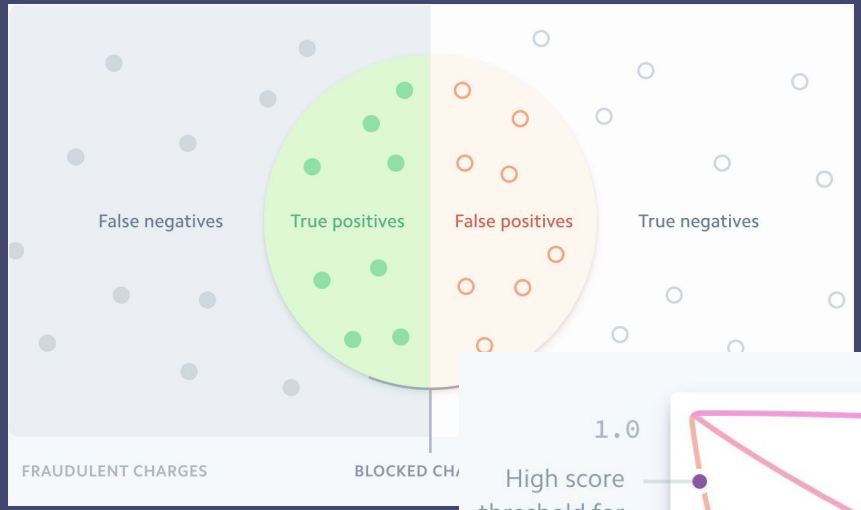
a	b	c	d	e	X
0	1	1	0	0	G
1	1	1	0	1	B
1	0	1	1	0	B
0	1	0	1	0	B
1	0	1	0	0	B



stripe


False negatives
True positives
False positives
True negatives
FRAUDULENT CHARGES
BLOCKED CHARGES
LEGITIMATE CHARGES

stripe


False negatives
True positives
False positives
True negatives
FRAUDULENT CHARGES
BLOCKED CHA


1.0
High score threshold for blocking
Precision
0.0
Great
Good
Okay
Low score threshold for blocking
0.0
Recall
1.0

stripe

Model training system wishlist

- Easy to get started
- Flexible - facilitate experimentation with libraries, model types, parameters
- Automatable
- Tracking and reporting
- Interfaces with ML ecosystem (e.g. features, inference)
- Reliable
- Secure
- Abstract away resource management

stripe

Model training system wishlist

Railyard API

- Easy to get started
- Flexible - facilitate experimentation with libraries, model types, parameters
- Automatable
- Tracking and reporting
- Interfaces with ML ecosystem (e.g. features, inference)
- Reliable
- Secure
- Abstract away resource management

Railyard on Kubernetes

stripe

Railyard API

stripe

stripe API

Find anything /

API Reference

The Stripe API is organized around REST. Our API has predictable resource-oriented URLs, accepts form-encoded request bodies, returns JSON-encoded responses, and uses standard HTTP response codes, authentication, and verbs.

You can use the Stripe API in test mode, which does not affect your live data or interact with the banking networks. The API key you use to authenticate the request determines whether the request is live mode or test mode.

The Stripe API differs for every account as we release new versions and tailor functionality. Log in to see docs customized to your version of the API, with your test key and data.

Subscribe to Stripe's API announce mailing list for updates.

Was this section helpful? Yes No

NOT A DEVELOPER?

Use apps from our partners to get started with Stripe and to do more with your Stripe account—no code required.

BASE URL

```
https://api.stripe.com
```

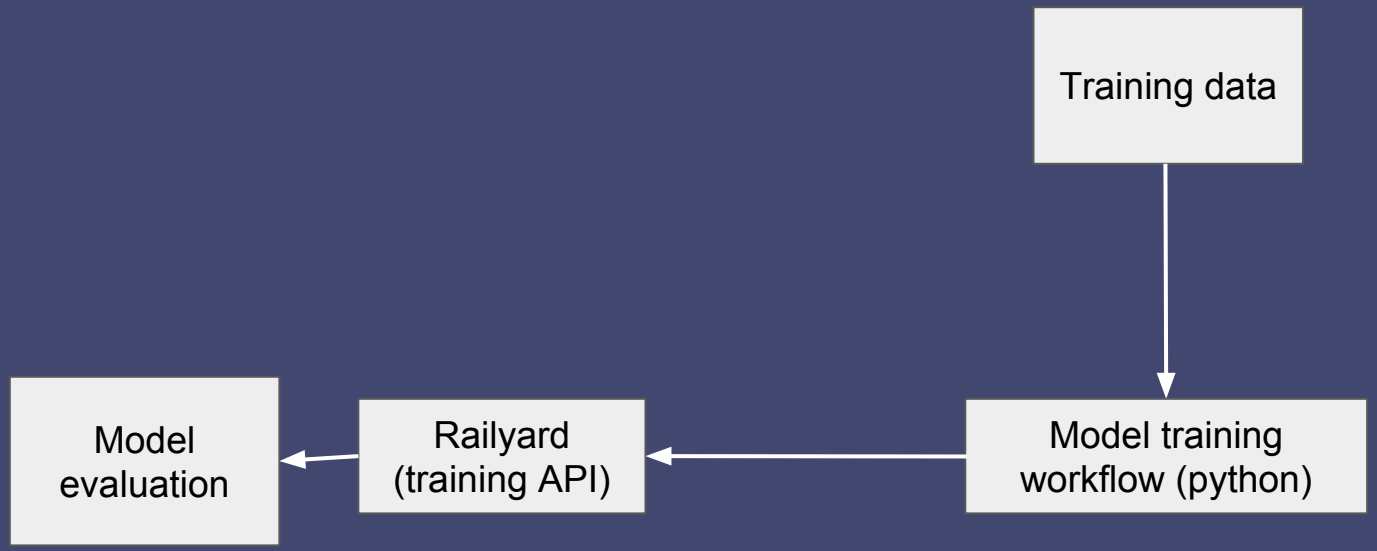
CLIENT LIBRARIES

Ruby Python PHP Java Node.js Go .NET

By default, the Stripe API Docs demonstrate using curl to interact with the API over HTTP. Select one of our official client libraries to see examples in code.

stripe

How it works



stripe

Example workflow

```
class StripeFraudModel(StripeMLWorkflow):

    def train(self, training_dataframe, holdout_dataframe):
        pipeline = Pipeline([
            ('boosted', xgboost.XGBRegressor(**self.custom_params))
        ])
        serializable_pipeline = stripe_ml.make_serializable(pipeline)
        fitted_pipeline = pipeline.fit(training_dataframe,
self.classifier_label)
        return fitted_pipeline
```

stripe

API Request: Metadata

```
{
  "model_description" : "A model to predict fraud",
  "model_name" : "fraud_prediction_model",
  "owner" : "machine-learning-infrastructure",
  "project": "strata-data-talk",
  "trainer": "kelley",
  ...
```

stripe

API Request: Data

```
"data" : {

 "features" : [

   {

     "names" : ["created_at", "charge_type", "charge_amount",
"charge_country", "has_fraud_dispute"],

     "path": "s3://path/to/parquet/fraud_data.parq"

   }

 ],

 "date_column": "created_at",
```

stripe

API Request: Filters

```
"filters" : [
{
  "feature_name" : "charge_country",
  "predicate" : "IsIn",
  "feature_value" : {
    "string_vals": ["US", "CA"]
  }
}],
```

stripe

API Request: Holdout data

```
"holdout_sampling" : {
  "sampling_function" : "DATE_RANGE",
  "date_range_sampling" : {
    "date_column" : "created_at",
    "start_date": "2018-10-01",
    "end_date": "2019-01-01"
  }
}
```

stripe

API Request: Training!

```
"train" : {
  "workflow_name" : "StripeFraudModel",
  "classifier_features": ["charge_type", "charge_amount"],
  "label" : "has_fraud_dispute"
  "custom_params": {
      "objective": "reg:linear",
      "max_depth": 6,
      "n_estimators": 500,
    }
```

stripe

API Request: Training!

```
"train" : {
  "workflow_name" : "StripeFraudModel",
  "classifier_features": ["charge_type", "charge_amount"],
  "label" : "has_fraud_dispute"
  "custom_params": {
      "objective": "reg:linear",
      "max_depth": 6,
      "n_estimators": 500,
    }
```

stripe

Example request and response

```
POST /train <request>
```

```
"9081e64f-b2c0-455e-bcaa-c1c211fa124b"
```

```
GET /job/{job_id}/status
```

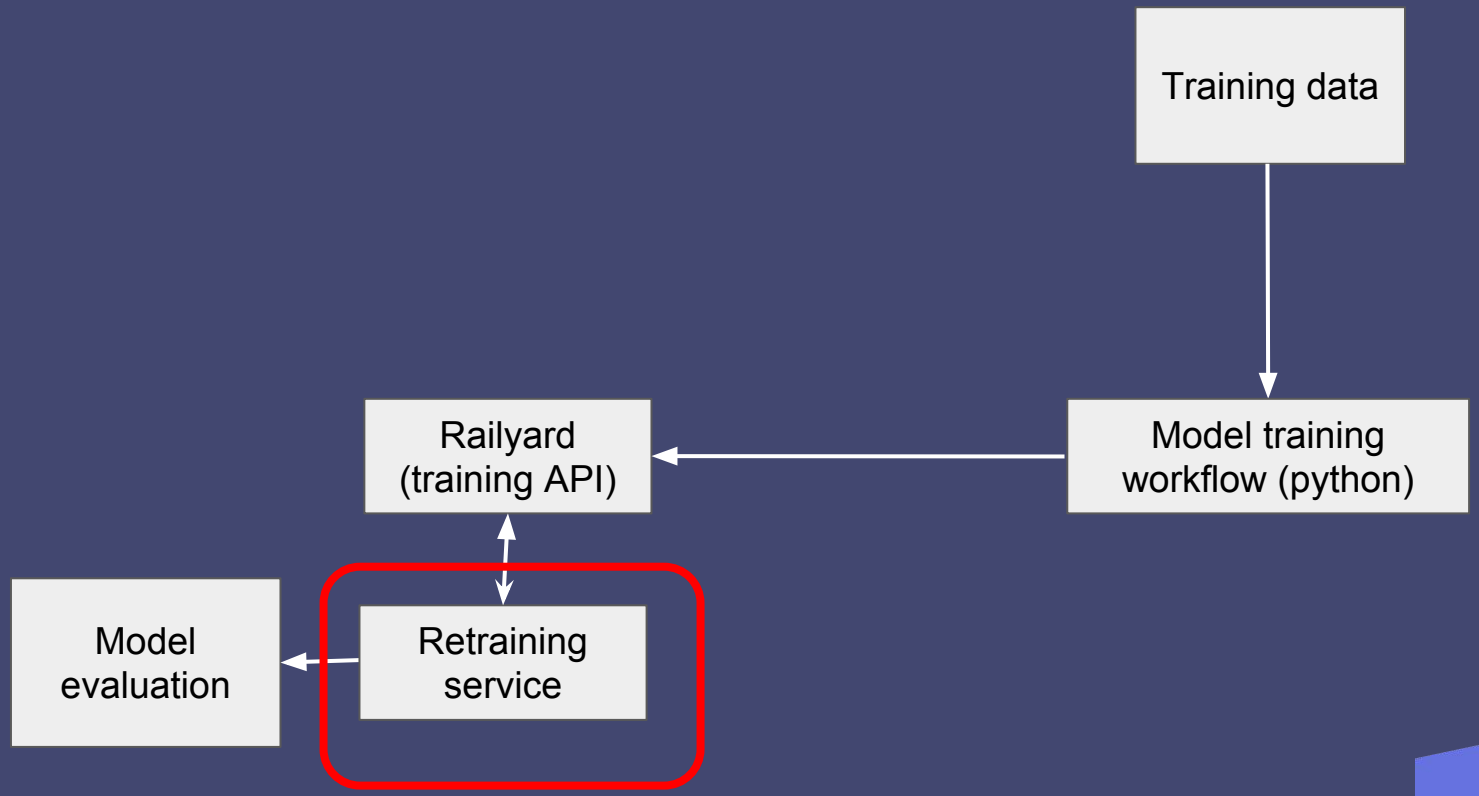
```
GET /job/{job_id}/result
```

stripe

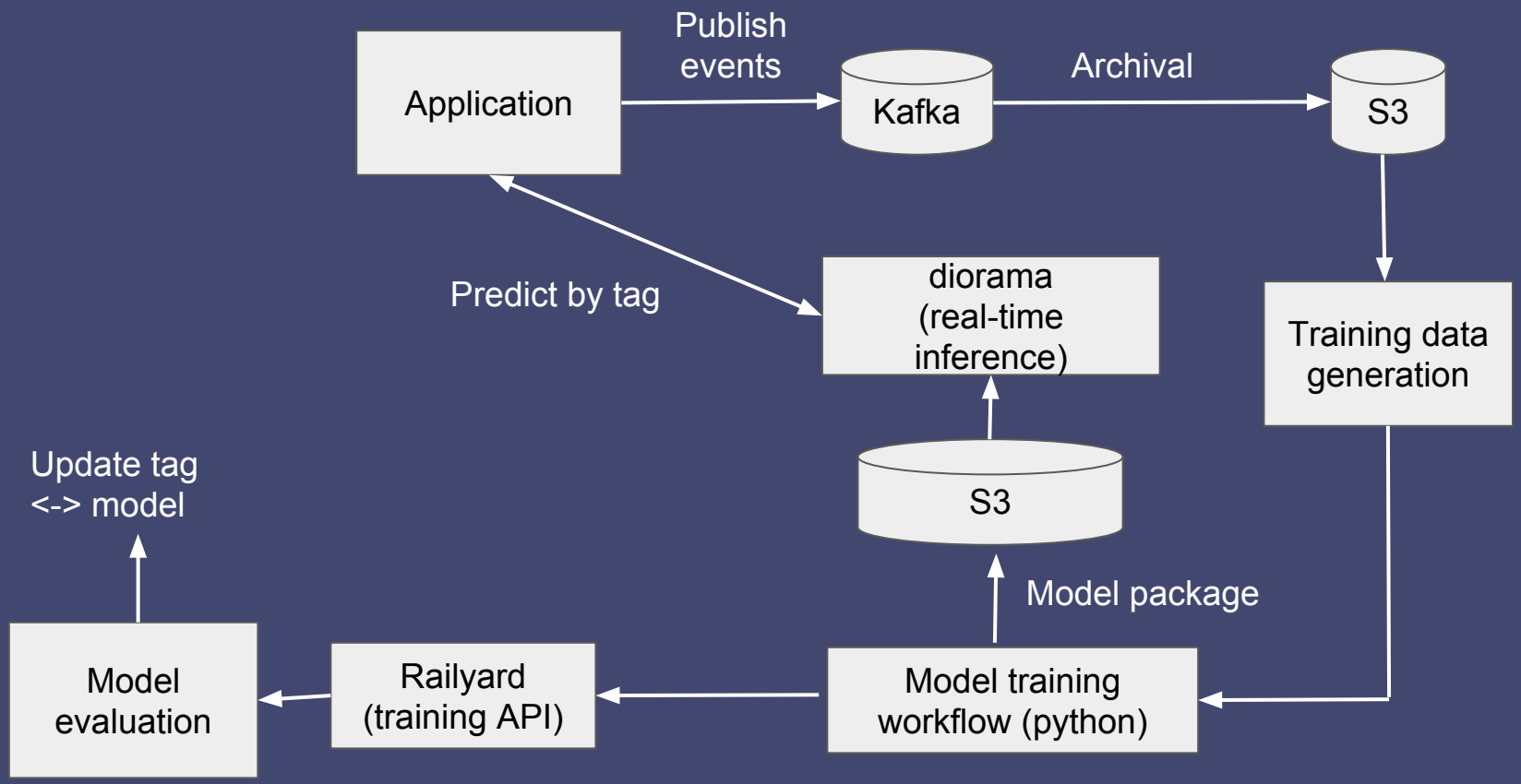
```
GET /job/{job_id}/result

{
  "status": {
    "job_id": {job_id},
    "log_file":"s3://{path}/{job_id}/logs",
    "transition": {
      "created_at":"2019-03-22 18:00:04 +0000",
      "job_state":"complete"
     },
     "git_commit":{git_SHA}
     },
  "result":{
    "evaluation_holdout_data_path":"s3://{dir}/{model_id}/scores.tsv",
    "evaluation_holdout_label_path":"s3://{dir}/{model_id}/labels.tsv",
    "diorama_id":"sha256.FDK2WAU4ULUV7ERWP3BMSVGPBGWG2GPUTUZXHOZRVSNCA4LPGVRA"
  },
  "exceptionInfo":null
  }
```

stripe

How it works



stripe


Application
Publish events
Kafka
Archival
S3
Predict by tag
diorama (real-time inference)
Training data generation
Update tag <-> model
S3
Model package
Model evaluation
Railyard (training API)
Model training workflow (python)

stripe

What we learned

API:

- Be flexible with model parameters
- Not using a DSL was the right choice for us.
- Tracking model provenance and ownership is really important

Workflow:

- Interfaces are important
- Users should not have to think about model serialization or persistence
- Measure each step

stripe

Model training system wishlist

Railyard API

- ✓ Easy to get started
- ✓ Flexible - facilitate experimentation with libraries, model types, parameters
- ✓ Automatable
- ✓ Tracking and reporting
- ✓ Interfaces with ML ecosystem (e.g. features, inference)

- Reliable
- Secure
- Abstract away resource management

Railyard on Kubernetes

stripe

Railyard on Kubernetes

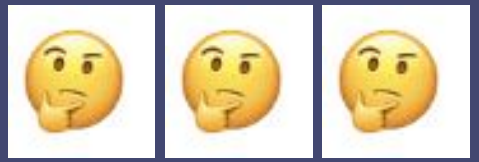
stripe

In the beginning

i3.16xlarge	i3.16xlarge	p3.2xlarge
sally	sally	mindy
	jim	joe
		sally

stripe

In the beginning

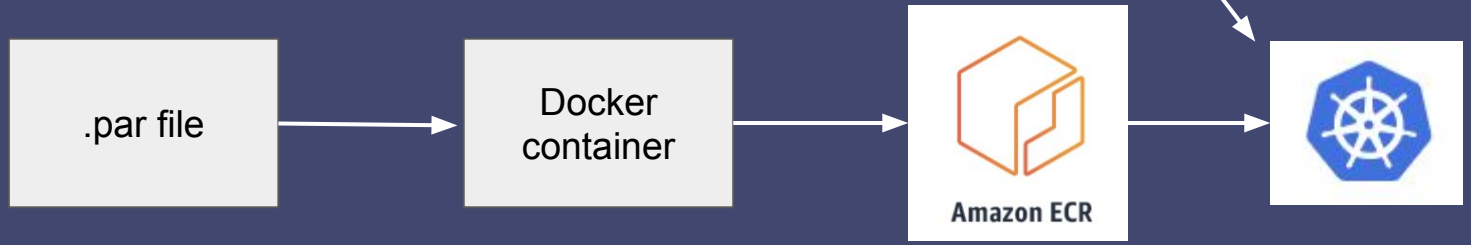


i3.16xlarge	i3.16xlarge	p3.2xlarge
sally	sally	mindy
	jim	joe
		sally

stripe

Running on Kubernetes

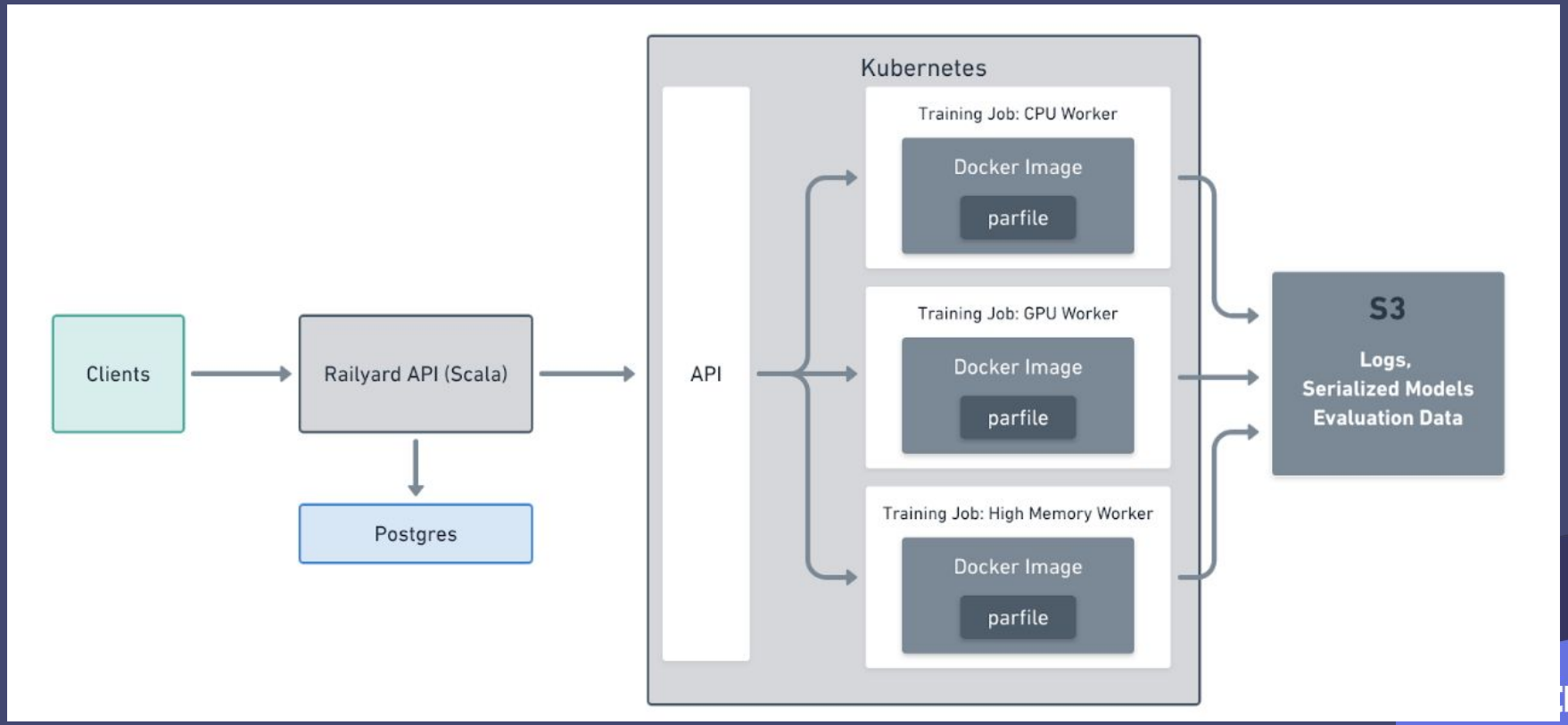
```
command: ["sh"]
args: ["-c", "python /railyard_train.par"]
```




```
par_binary(
    name = "railyard_train",
    srcs = ["@.../ml:railyard_srcs"],
    data = ["@.../ml:railyard_data"],
    main = "@.../ml:railyard/train.py",
    deps = all_requirements,
)
```

stripe

Running on Kubernetes



Heterogeneous workflows

```
{
    "compute_resource": "GPU"
}
```

stripe

Model training system wishlist

Railyard API

- ✓ Easy to get started
- ✓ Flexible - facilitate experimentation with libraries, model types, parameters
- ✓ Automatable
- ✓ Tracking and reporting
- ✓ Interfaces with ML ecosystem (e.g. features, inference)

- ✓ Reliable
- ✓ Secure
- ✓ Abstract away resource management

Railyard on Kubernetes

stripe

What we learned

- Instance flexibility is important!
- Still takes some trial and error
- Subpar was a great choice for us
- Having a good Orchestration team running Kubernetes has been a force multiplier.

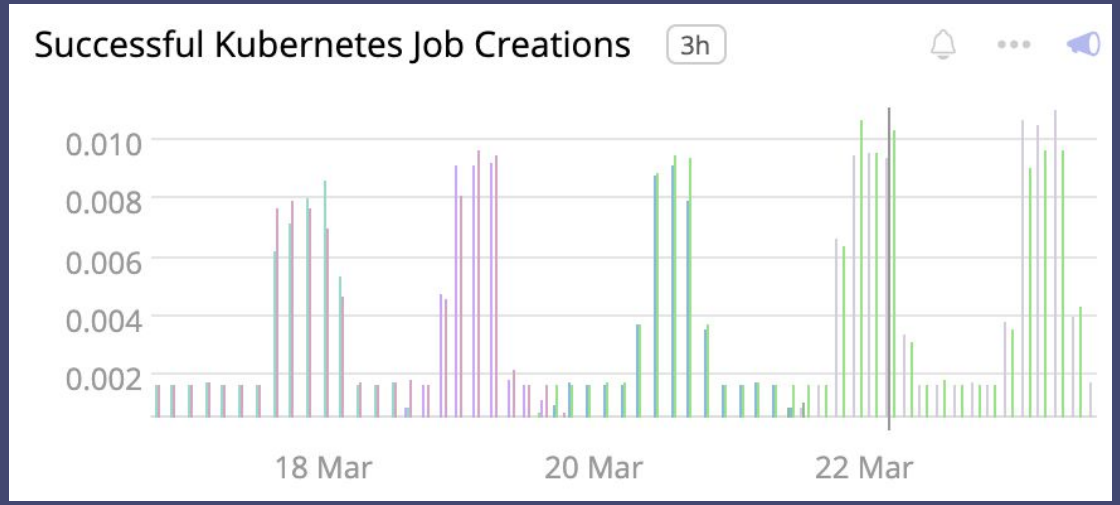
stripe

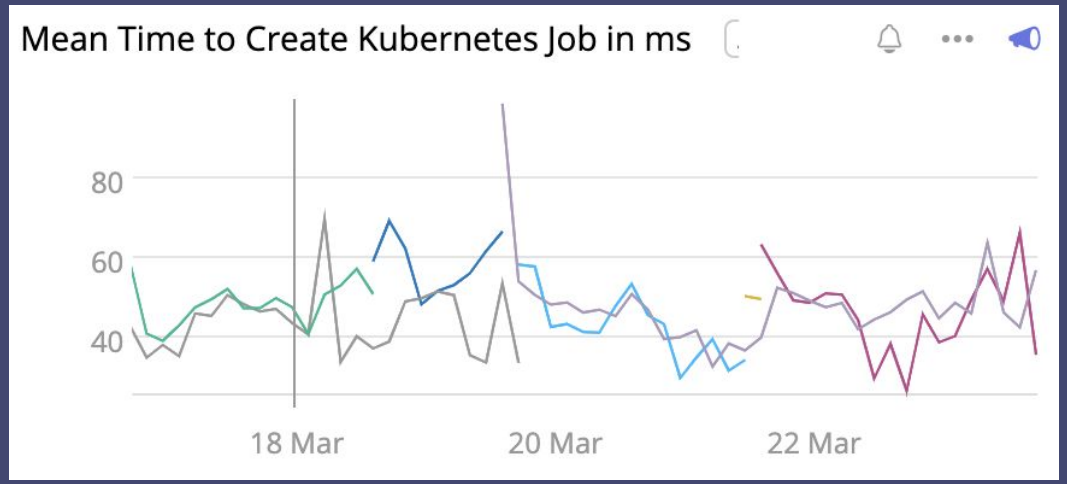
Railyard in action

stripe

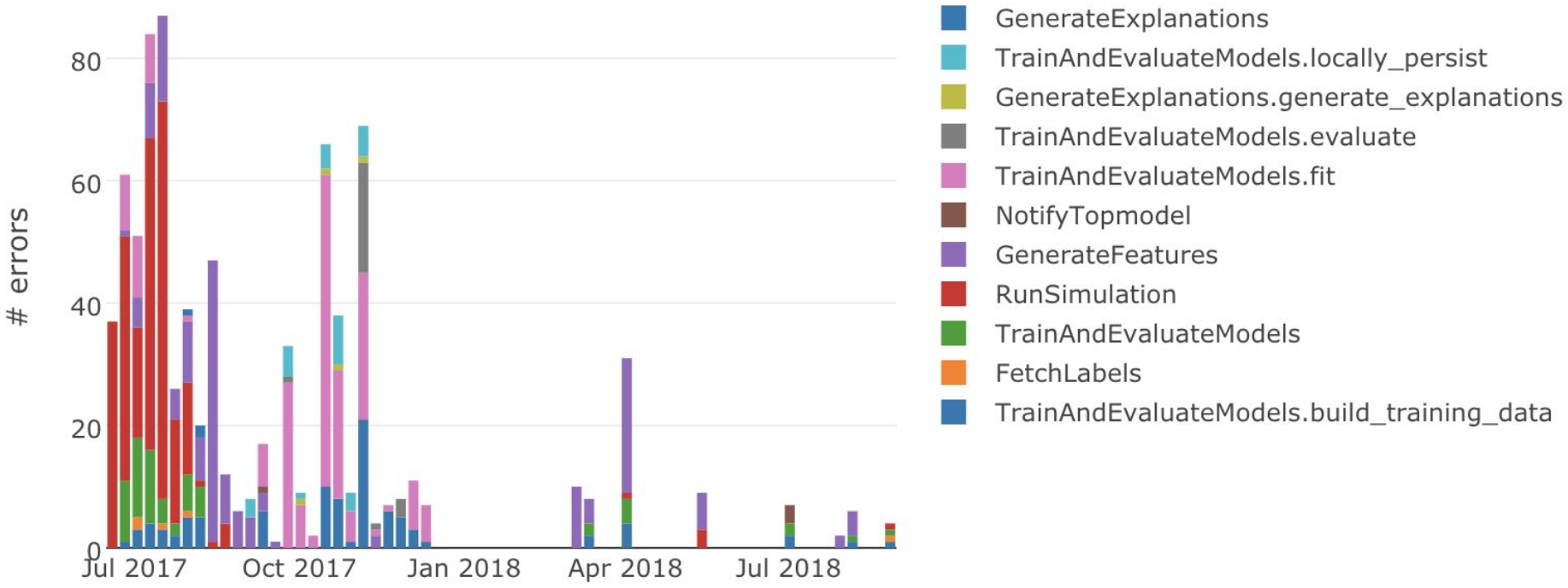
2019-03-21 00:45:00 UTC	50 minutes	NOT SHIPPED	training	
2019-03-21 00:45:00 UTC		RUNNING	training	
2019-03-21 00:43:59 UTC	25 minutes	FAILED	training	`java.lang.Exception: Railyard training failed with exit value: 1.` `Error: Some(AssertionError: There should be at least 500 examples`
2019-03-21 00:43:06 UTC	7 minutes	FINISHED	explanation model	
2019-03-21 00:42:58 UTC	57 minutes	SHIPPED	training	
2019-03-21 00:38:53 UTC	47 minutes	SHIPPED	training	
2019-03-21 00:37:39 UTC	26 minutes	FINISHED	explanation model	

stripe


Successful Kubernetes Job Creations
3h
0.010
0.008
0.006
0.004
0.002
18 Mar
20 Mar
22 Mar


Mean Time to Create Kubernetes Job in ms
80
60
40
18 Mar
20 Mar
22 Mar

stripe


errors
80
60
40
20
0
Jul 2017
Oct 2017
Jan 2018
Apr 2018
Jul 2018
GenerateExplanations
TrainAndEvaluateModels.locally_persist
GenerateExplanations.generate_explanations
TrainAndEvaluateModels.evaluate
TrainAndEvaluateModels.fit
NotifyTopmodel
GenerateFeatures
RunSimulation
TrainAndEvaluateModels
FetchLabels
TrainAndEvaluateModels.build_training_data

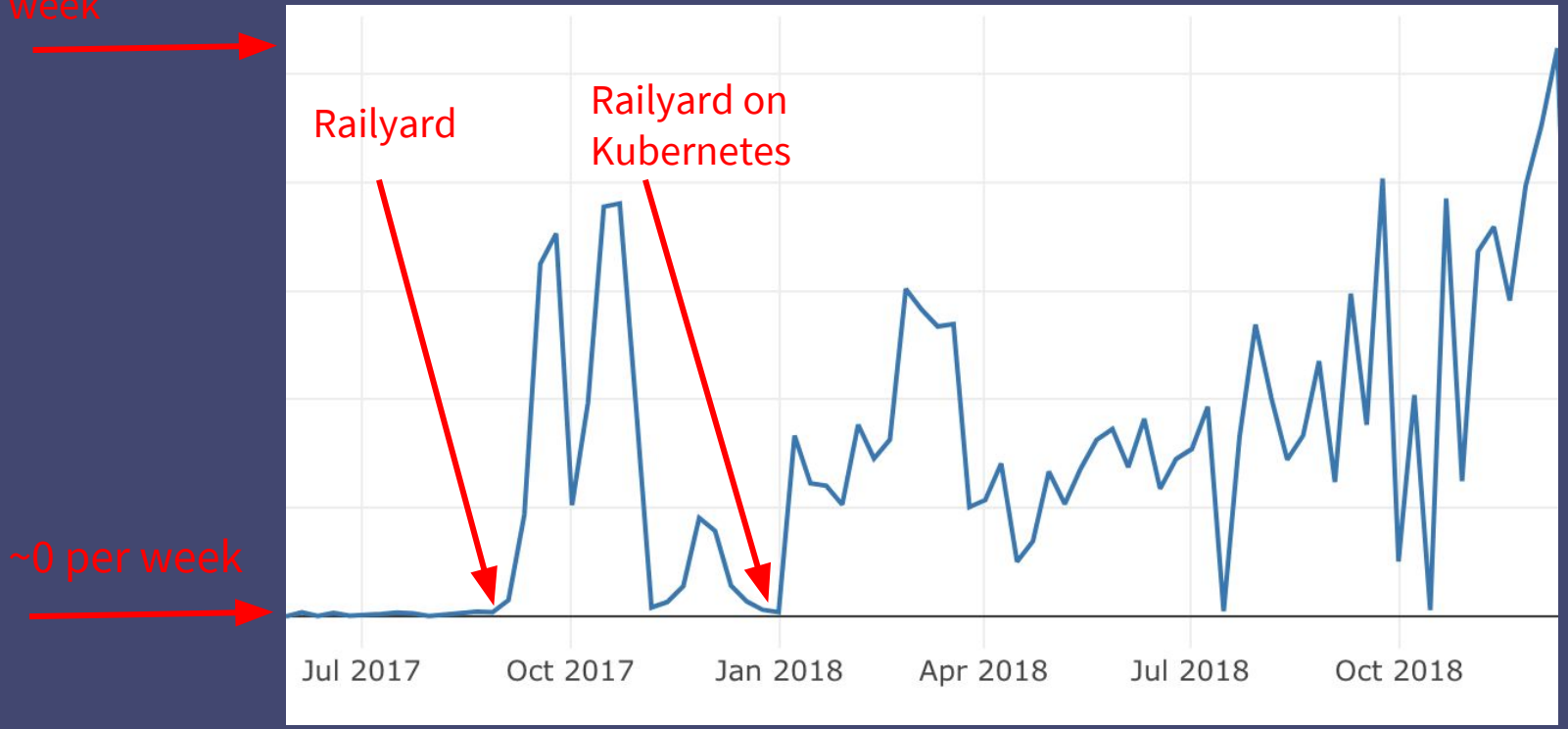
stripe

By the numbers

- Many workflows - from user-facing products like Radar to payments optimization to internal-facing modeling and risk management
- Libraries including scikit-learn, pytorch, fasttext, xgboost, and prophet
- *Hundreds of thousands* of models trained, *thousands more every week*
- CPU, GPU, and high memory resource types
- Models used in *100s of millions* of real-time predictions every day

stripe

Number of models trained



stripe

What we did

- Simple but flexible API for running and automating training workflows
- Resource management via Kubernetes to reduce toil, improve reliability and security
- Instrumentation throughout to track model provenance and ownership, as well as debug and profile training jobs
- We use it to train thousands of models per week for a range of user-facing and internal ML applications

stripe

Feedback from our users!

“Training models with railyard has been nice - it’s saved me time by abstracting away the more tedious parts of training (loading data, separating training/test sets, fitting and scoring, writing output files), allowing me to focus more on building features and model architecture.”

“Railyard has made it much simpler to write a new pipeline. When <new teammate> started, I was able to simply point him towards docs to get him going.”

“I explained the ml stack for <my project> to several people on <my> team and they were really relieved to hear that training code used a "standard" way of doing things that they could count on others knowing about.”

stripe

Thanks / come work with me :)

- Stripe is hiring for interesting Data roles in Seattle, SF, and remote, using data to track and move money, build state-of-the-art ML

- Special thanks to Rob Story, Thomas Switzer, and Sam Ritchie

stripe