Дизайн систем машинного обучения

11. Эксперименты и версионирование

План курса

1) ~~Практическое применение машинного обучения~~
2) ~~Основы проектирования ML-систем~~
3) ~~Обучающие данные~~
4) ~~Подготовка и отбор признаков~~
5) ~~Выбор модели, разработка и обучение модели~~
6) ~~Оценка качества модели~~
7) ~~Развертывание~~
8) ~~Диагностика ошибок и отказов ML-систем~~
9) ~~Мониторинг и обучение на потоковых данных~~
10) ~~Жизненный цикл модели~~
11) **Отслеживание экспериментов и версионирование моделей — Вы находитесь здесь**
12) Сложные модели: временные ряды, модели над графами
13) Непредвзятость, безопасность, карточки моделей
14) ML инфраструктура и платформы
15) Интеграция ML-систем в бизнес-процессы

Что можно отслеживать

- Данные → ?
- Код предобработки данных → git
- Код обучения модели → git
- Веса модели → ?
- Метрики модели → ?
- Предсказания модели → ?
- Логи / журналы обучения → ?

Версионирование данных

- Модели имеют смысл только в контексте данных
- Данных сильно больше, чем кода.
- Мы не можем версионировать данные так же, как делаем это с кодом
- Трудно делать дифы. Трудно читать эти дифы. Трудно хранить все изменения датасета, особенно если он большой
- Датасет может не помещаться на локальной машине
- *...at this point, data versioning is like flossing.*
  Everyone agrees it's a good thing to do, but few do it © Chip Nguen

Что такое «изменение данных»

- Изменение схемы данных ломает модель
- Отслеживать и проверять схему данных крайне желательно.
- Нужна ли нам версионность на самих данных?
  - Копия данных при каждом обучении модели
  - Версионность кода извлечения данных
  - Версионность данных на уровне модели данных

Как быть с «изменениями данных»

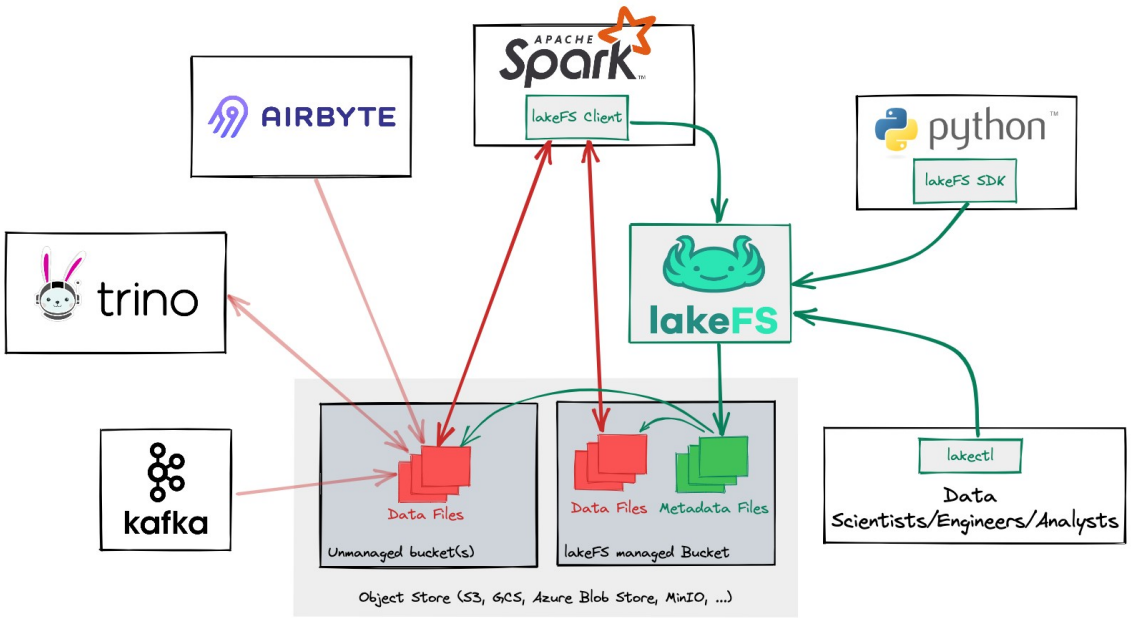
- Что считать изменившимися данными?
- Контрольная сумма — а если они в базах?
- Возможно ли сливать изменения?
- Иногда хранить данные в репозитории нельзя (как и токены с паролями— это небезопасно)
- Их много

Версионирование данных

- Можно организовать хранение данных так, чтобы история изменения данных сохранялась
- Event Sourcing
- Anchor Modeling
- Slowly Changing Dimensions
- Нужно версионировать код извлечения данных
- Но об этом нужно было думать заранее

https://habr.com/ru/company/avito/blog/322510/

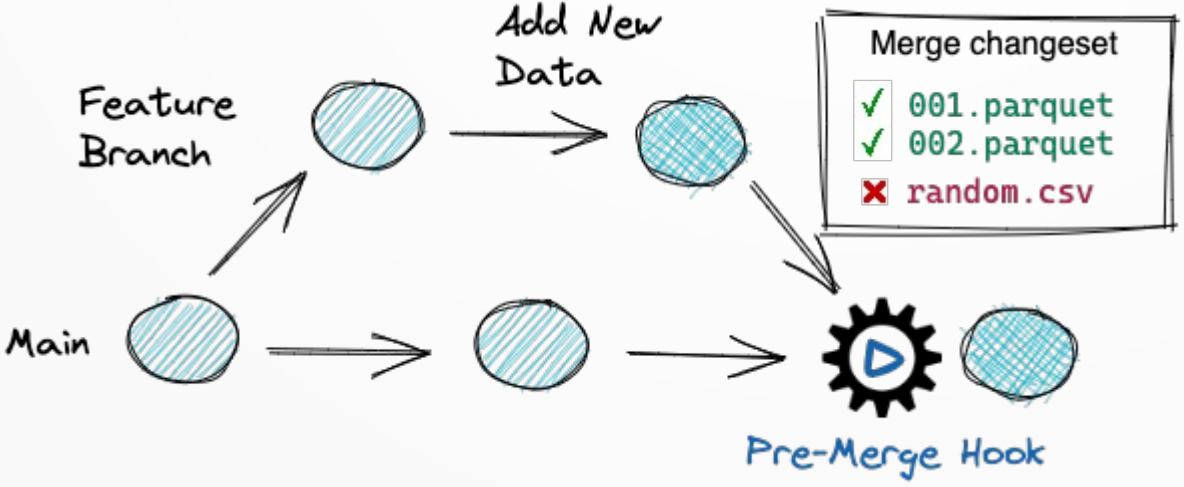
LakeFS — версионирование S3



https://docs.lakefs.io/

LakeFS - слияние

- Branch + Merge для данных, как в git



https://docs.lakefs.io/

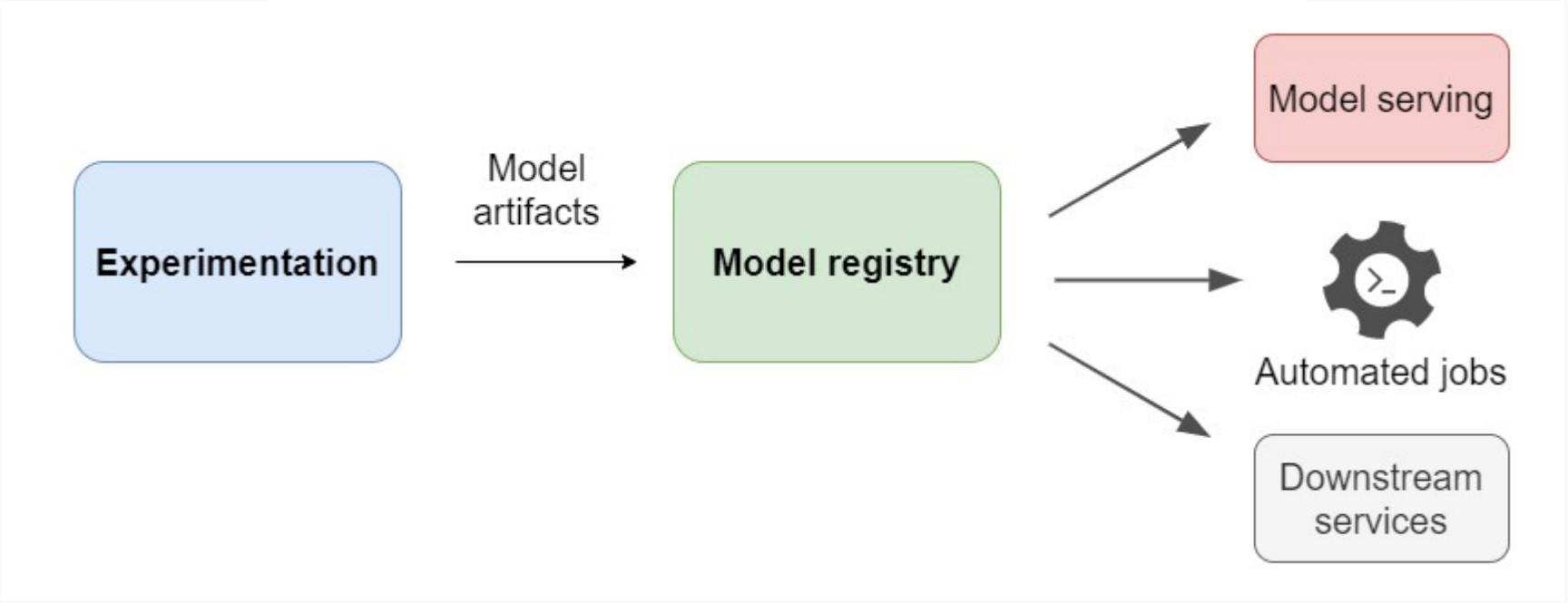
Отслеживание экспериментов

- Кривая потерь
- Метрики качества модели
- Ссылка на данные, на которых модель обучалась
- Скорость работы модели (точек данных в секунду)
- Метрики системы (память, утилизация CPU, GPU и т.д.
- Значения параметров и гиперпараметров модели

Версионирование моделей

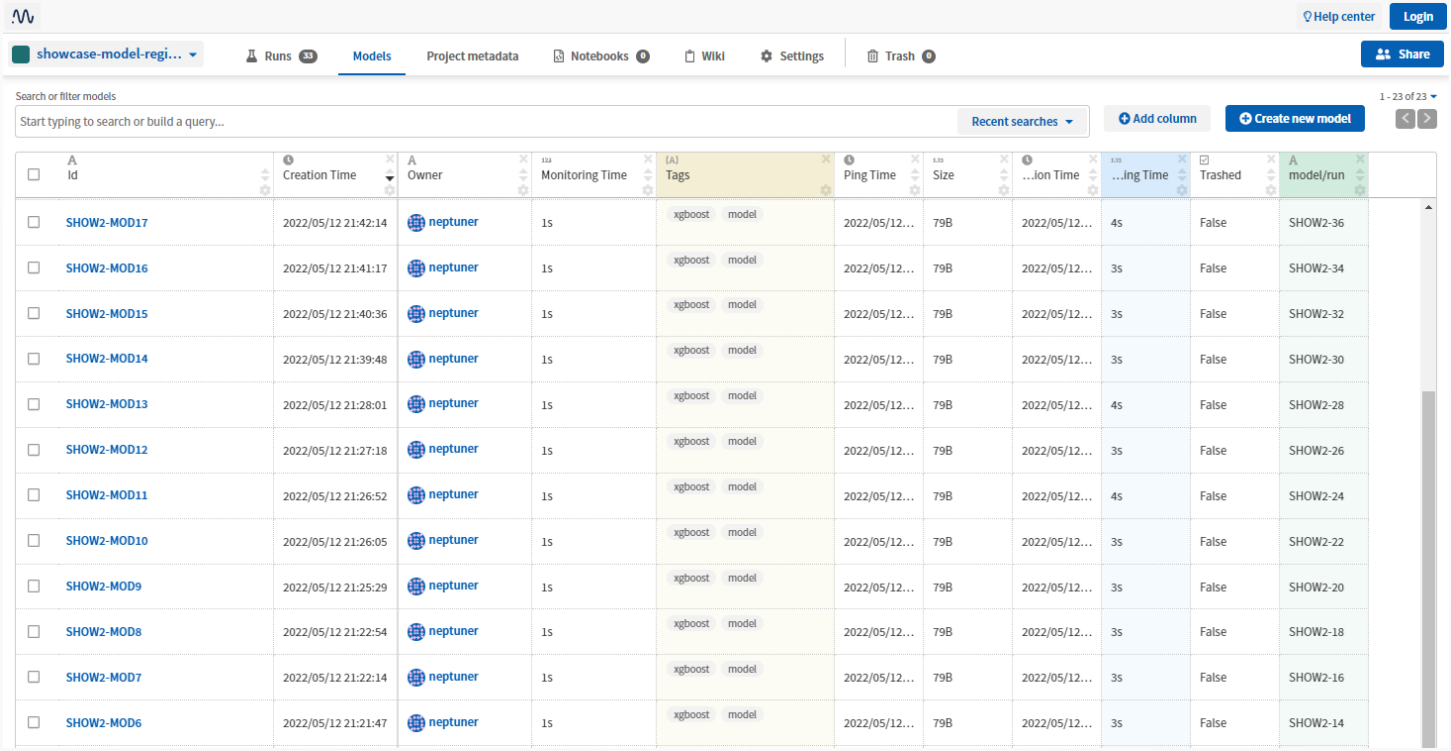
- Когда вы пишете код и ломаете его, вы можете с помощью системы контроля версий вернуть его в прошлое, работоспособное состояние
- Можете ли вы сделать то же самое с моделью машинного обучения? Отменить часть изменений, отследить — какие из них ухудшили качество, вернуться к одной из прошлых удачных версий и продолжить работу от этой точки?
- Обычно мы можем отслеживать только код обучения

Model Registry - NeptuneAI



https://neptune.ai/product/model-registry

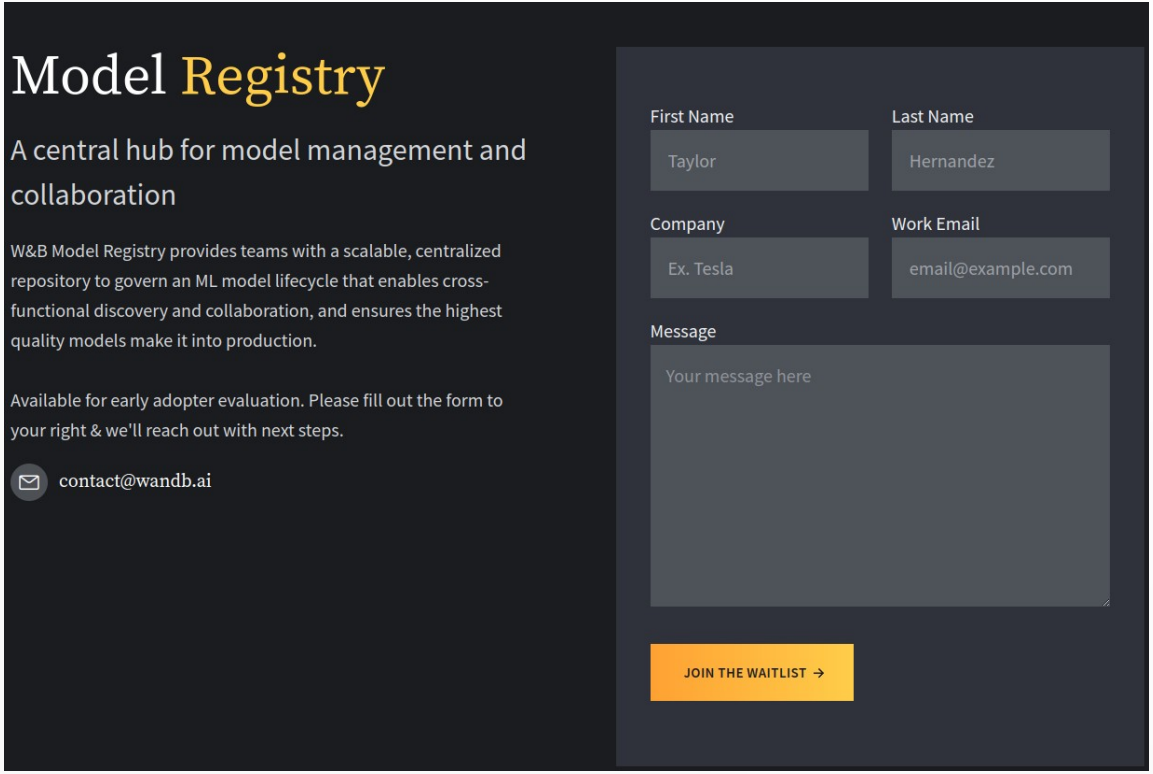
Model Registry - NeptuneAI



Id	Creation Time	Owner	Monitoring Time	Tags	Ping Time	Size	...ion Time	...ing Time	Trashed	model/run
SHOW2-MOD17	2022/05/12 21:42:14	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-36
SHOW2-MOD16	2022/05/12 21:41:17	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-34
SHOW2-MOD15	2022/05/12 21:40:36	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-32
SHOW2-MOD14	2022/05/12 21:39:48	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-30
SHOW2-MOD13	2022/05/12 21:28:01	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-28
SHOW2-MOD12	2022/05/12 21:27:18	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-26
SHOW2-MOD11	2022/05/12 21:26:52	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	4s	False	SHOW2-24
SHOW2-MOD10	2022/05/12 21:26:05	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-22
SHOW2-MOD9	2022/05/12 21:25:29	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-20
SHOW2-MOD8	2022/05/12 21:22:54	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-18
SHOW2-MOD7	2022/05/12 21:22:14	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-16
SHOW2-MOD6	2022/05/12 21:21:47	neptuner	1s	xgboost model	2022/05/12...	79B	2022/05/12...	3s	False	SHOW2-14

https://neptune.ai/product/model-registry

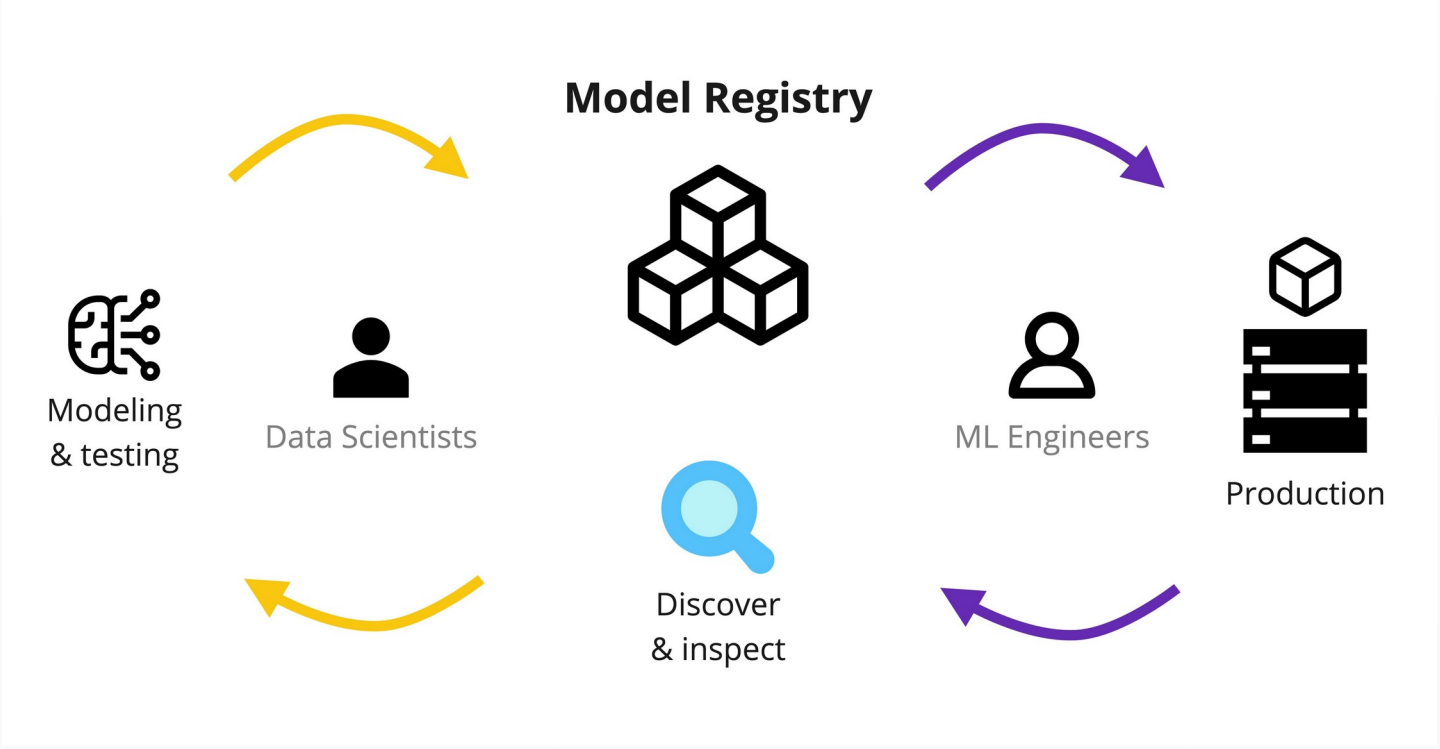
W&B — wait a minute



https://wandb.ai/site/models

W&B выкатили реестр моделей, но это было уже после записи лекции ;-)

DVC Model Registry



https://dvc.org/doc/use-cases/model-registry

Git as Model Registry

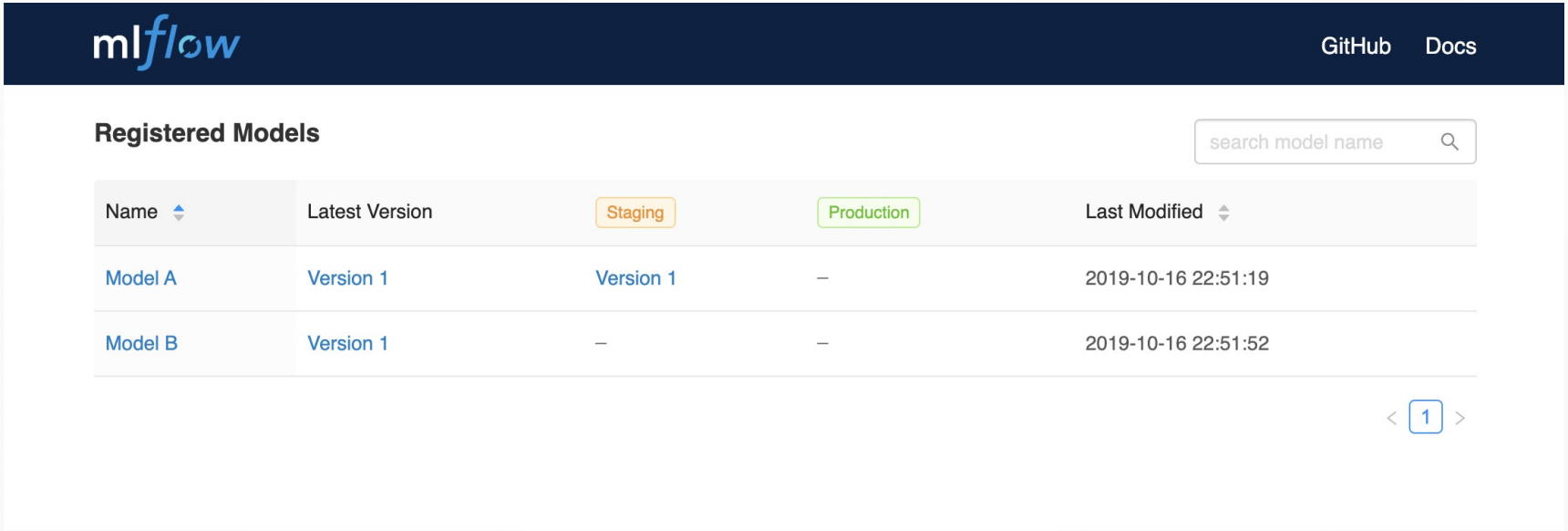
- Git Tag Ops →
- MLEM →

```
76% |████████▌     | 7568/10000
100%|██████████████| 10000/10000
>>>  import mlem
>>>  mlem.api.save(model, "dog-bark-translator")
>>>

$  tree
.
├── dog-bark-translator
└── dog-bark-translator.mlem

$
```

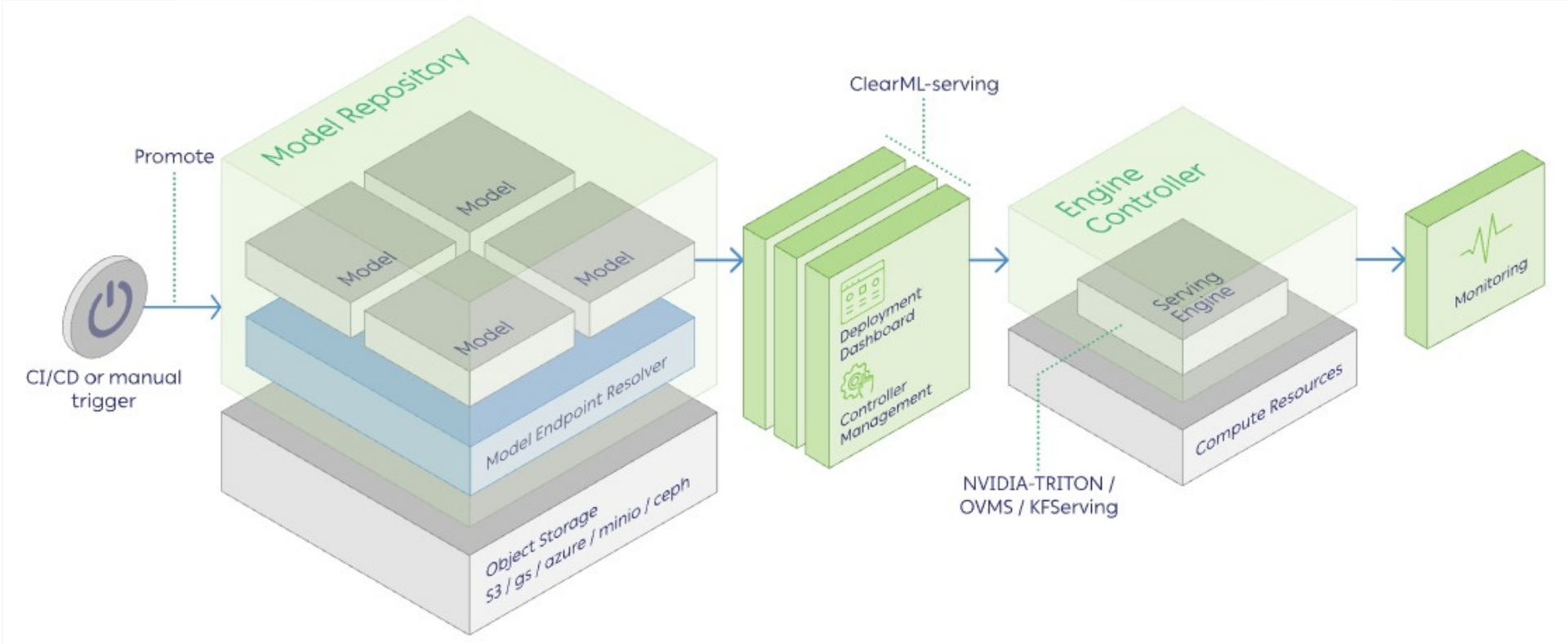
MLFlow



https://learn.microsoft.com/ru-ru/azure/databricks/mlflow/model-registry-example

https://www.mlflow.org/docs/latest/model-registry.html

ClearML



https://clear.ml/products/clearml-deploy/

В итоге

- Аггресивное отслеживание экспериментов и версионирование помогает с воспроизводимостью
- Но не гарантирует ее.
- У библиотек могут быть проблемы с детерменизмом
- Оборудование может вносить недетерменизм в модели
- Данные могут быть невоспроизводимо меняться
- Разные модели GPU могут считать по-разному

Random seed

- Seed = 20221117 # YYYYMMDD

```
import torch
torch.manual_seed(0)
```

```
import random
random.seed(0)
```

```
import numpy as np
np.random.seed(0)
```

- WARNING

Deterministic operations are often slower than nondeterministic operations, so single-run performance may decrease for your model. However, determinism may save time in development by facilitating experimentation, debugging, and regression testing.

https://pytorch.org/docs/stable/notes/randomness.html

Воспроизводимость не всегда возможна

• WARNING

There are known non-determinism issues for RNN functions on some versions of cuDNN and CUDA.
You can enforce deter

On CUDA 10.1, set env

On CUDA 10.2
CUBLAS_WORKS

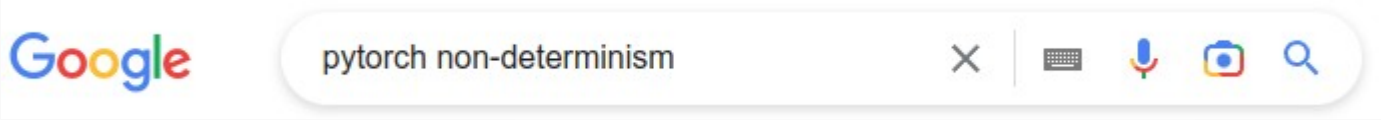
See the cuDNN

If you are calling `backward()` on multiple thread concurrently but with shared inputs (i.e. Hogwild CPU training). Since parameters are automatically shared across threads, gradient accumulation might become non-deterministic on backward calls across threads, because two backward calls might access and try to

and

Common Pitfalls in Transform Authoring

- Nondeterministic `set` iteration order. In Python, the `set` datatype is unordered. Using `set` to contain collections of objects like `Node`s, for example, can cause unexpected nondeterminism. An example is iterating over a set of `Node`s to insert them into a `Graph`. Because the `set` data type is unordered, the



https://github.com/pytorch/pytorch/issues?q=is%3Aissue+is%3Aopen+nondeterminism

Вместо чтения статей

- W&B QuickStart
- DVC Get Started
- MLFlow Quickstart
- ClearML First Step
- Tensorboard Get Started

- Попробуйте каждый из этих инструментов