kolesa group

Что я хотел бы знать об MLOps год назад

Миржан Иркегулов,
Backend-разработчик, ML & Operations

Завязка

- Знакомься, твоя новая команда
  - 3 ML-инженера
  - 2 MLOps-инженера
- Все пришли в 2022 году

Legacy

- 10–15 сервисов
- ~50 моделей
- Python + PyTorch

Постановка проблемы

- Компания завязалась на ML-сервисы
- Растёт time-to-market (TTM)
- Бэклог размером с год
- Всё труднее поддерживать
- Постоянные алерты и поломки

Идея-фикс:
снизить TTM

Кейсы

- Кейс №1: Реестр моделей
- Кейс №2: Воспроизводимость (reproducibility)
- Кейс №3: Зоны ответственности
- Кейс №4: GPU

Устройство кейсов

1. Как было до
2. Проблема
3. Решение
4. Как стало после

Кейс №1: реестр моделей

Как было до: модели

- PyTorch: deep learning framework
- Модель: обученная нейросеть
- Сериализация: .pth
- Модели хранятся в Ceph
  - Распределённое хранилище данных
  - Интерфейс Amazon S3
  - Интерфейс OpenStack

Проблема: хранение моделей

1. Модели размазаны по хранилищам
   a. Непонятно: какая модель у какого сервиса?
   b. 50 Гб моделей, не привязанных к сервисам

2. ML-инженер вручную загружает модель в Ceph
   a. Нет бэкапов
   b. Как откатиться?
   c. Нет single source of truth для моделей

Решение

- Нам нужен реестр моделей!
- Ищем MLOps-инструменты:
  - Занимают разные ниши
  - Решают разные задачи

- Они не взаимозаменяемы

MLOps-инструменты

- Cog
- Seldon
- Kubeflow
- Nvidia Triton
- MLflow ✅

- Простой веб-интерфейс
- Трекинг экспериментов
- Версионирование моделей
- Метрики
- Стандартизация workflow
- Flavor: интеграция с PyTorch

Кейс №2: воспроизводимость

Нейросеть нужно обучать

- Две стадии нейросети:
  - Training
  - Inference

- Для обучения нужны данные
  - Много данных

Модель без данных = 👎

- *No interpretability*: Нейросеть — это чёрный ящик
  - Нейросеть без данных ≍ машинный код
- Нет версионирования = нет метрик
  - Невозможно понять: правильно ли модель работает?
- *No reproducibility*: Невозможно переобучить
- *Model decay*: переобучать придётся!

Как было до: где данные?

- Часть лежит на машинах для тренировки
- Часть размазана по хранилищам и дата-центрам
- Даже если есть:
  - Неполные
  - Устаревшие

Версионирование данных

- Разные версии моделей = разные датасеты
- Данные тоже нужно версионировать

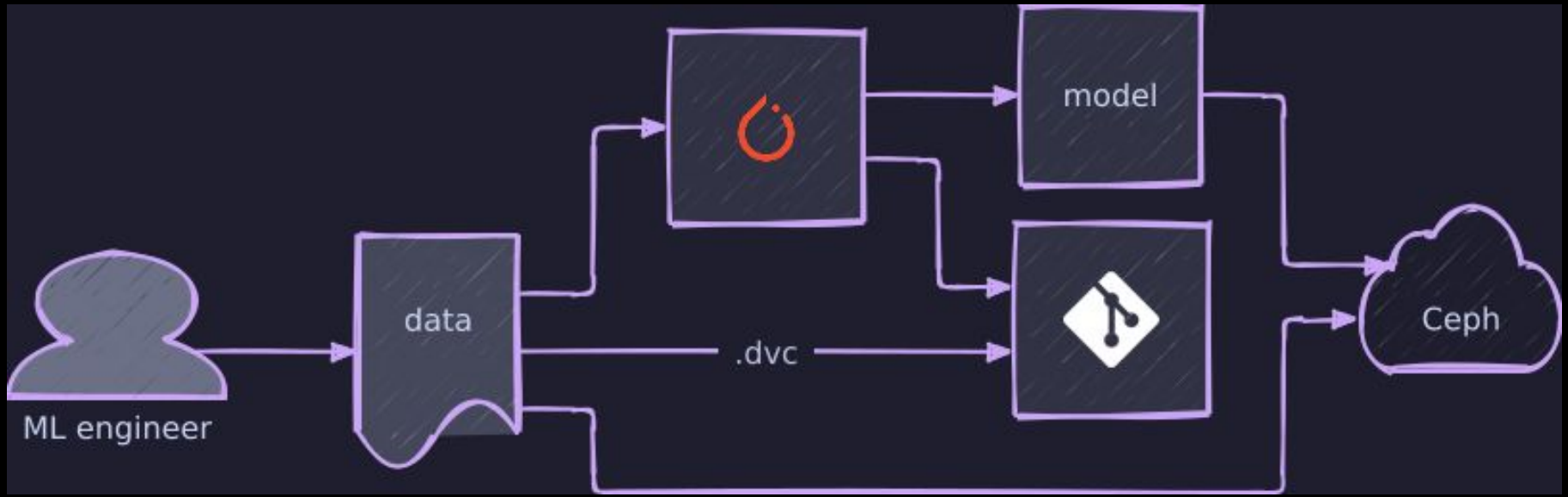
Где хранить данные?

- Данные нельзя хранить в Git
- Git LFS не был задуман для data science
- А где их хранить? Hadoop? S3?
- Так же, как и везде: Ceph

Решение: DVC

- Как Git, только для данных
- Данные хранятся удалённо в Ceph
- В Git хранятся метаданные
- Результат: файл .dvc
  - Его мы коммитим в Git

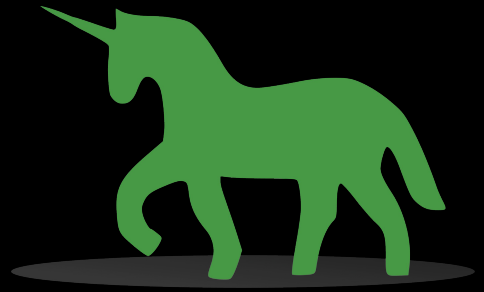
Как стало сейчас



Кейс №3:
зоны ответственности

Устройство ML-сервиса

- Модель (≥1)
- Нетривиальная бизнес-логика
- Зоопарк веб-фрейворков: Flask, FastAPI
- Зоопарк веб-серверов: Tornado, Gunicorn
- Endpoints
- Dockerfile
- CI/CD (Bamboo)
- Kubernetes



Обязанности ML-инженера

- Собрать данные, натренировать модель
- Реализовать релевантную бизнес-логику
- Написать микросервис с endpoints
- Написать Dockerfile
- Написать Bamboo build plan
- Написать Bamboo deployment project
- Написать манифесты для Kubernetes
- Настроить мониторинг

Мы физически не потянем

- Отдел Backend-разработки: ~50 человек
- Команда ML & Operations: 5 человек

Отдел Backend

Отдел ML

Bus factor

- Backend пишет на Go/PHP
- Backend возьмёт бизнес-логику на себя
- Нужно отвязаться от Python

MLflow для MLOps

- MLflow умеет оборачивать в сервис на базе Gunicorn
- Два стандартных endpoints:
  - /invocations
  - /health

- MLflow умеет оборачивать в Docker-контейнер и отдавать его

Но не тут-то было

- MLflow требует выделенный сервер
  - Вертится как Docker-контейнер в Kubernetes
  - Собирает Docker-образ локально
  - Docker-in-docker
  - Не хватает памяти!
- Выделенные агенты для сборки Docker
  - Не используются
- Неоптимально!

Сборка Docker: 😔

- Полученный Docker-образ не работает
- Не видит видеокарты!
- Вручную собираем образы: всё заработало!
- Наш ручной образ > образ MLflow
- MLflow: не отдавай образ, отдавай Dockerfile
- Мы открываем pull request:
  - https://github.com/mlflow/mlflow/pull/6591
  - Начиная с версии 1.29

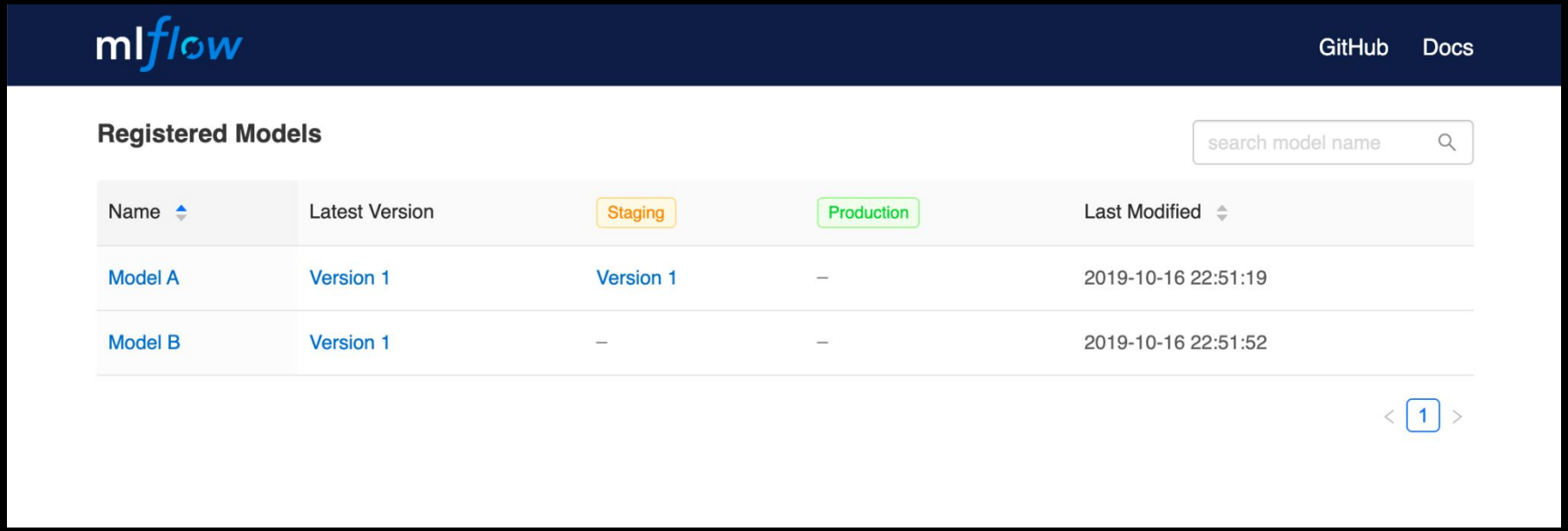
CI/CD

- Мы вручную отправляем контейнеры в Kubernetes
  - Какой же это DevOps?
- MLflow не понимает CI/CD
- MLflow не понимает Kubernetes
  - Манифесты
  - Объекты (Deployment, Ingress, Service, etc.)

MLflow + 💥 + CI/CD

- MLflow умеет всё *до* генерации Docker-контейнера
- Наш стэк CI/CD умеет всё *от* Git-репозитория
- Между MLflow и CI/CD разрыв
- Как их соединить?

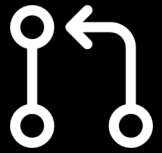
Реестр моделей: revisited



Прикрутить функционал

- Изменение в реестре: вебхук
- Генерируется манифест Kubernetes (YAML)
- Кладётся в Git-репозиторий

Мексиканская дуэль



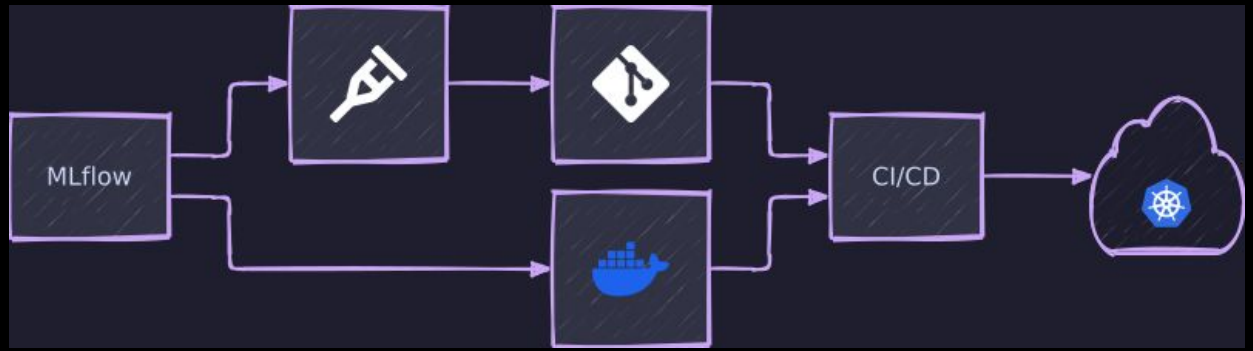
Pull request

Fork

Костыль

Костыль: MLflow CI/CD

- Подхватываем изменения в реестре моделей
- Генерируем манифесты Kubernetes
- Забираем Dockerfile у MLflow
- Собираем Docker-образ
- Деплоим на Kubernetes



Кейс №4: GPU

Причём здесь видеокарты?

- Видеокарта как parallel computing в миниатюре
- ML — это перемножение матриц из floating-point numbers
- Перемножение матриц отлично параллелезируется
- GPU быстрее CPU до 50 раз для ML
- GPGPU: General-purpose computing on GPU
    - NVIDIA CUDA

Головокружение от успехов

- TTM снизился
- Модели начинают клепаться быстро
- Слишком быстро?

Empire Strikes Back

- Пьём чай, прибегает разработчик из соседней команды
- Всё сломалось, ваш ML-сервис упал, сыплет логами
- Хорошо, откатим
- Падают старые сервисы, которые работали нормально
- Смотрим: кончилась память на видеокартах

Zabbix для GPU

- Все знают метрики (Graphite/Prometheus)
- Мало кто говорит про мониторинг самого железа
- План был
- Недооценили масштаб
- Выясняется: видеокарты нагружены неравномерно

Постановка проблемы

- Память GPU заканчивается
- Новое железо подвезут не скоро
- Убрать сервис нельзя
- Количество реплик: снизили как могли

Решение

1. Мораторий на новые релизы
2. Избежать резких скачков в потреблении
3. Масштабировать в ручном режиме, если придётся
4. Научиться сажать на видеокарты динамически

PyTorch

- PyTorch не умеет динамически выбирать видеокарту
- Ему доступную видеокарту нужно явно указать

```
tensor = torch.zeros(2,3)
if torch.cuda.is_available():
    tensor.to(torch.device("cuda:0"))
else:
    tensor.to(torch.device("cpu"))
```

Динамическая нагрузка

- Как научить PyTorch:
  - Не хардкодить номер видеокарты?
  - Выбирать наименее загруженную видеокарту?
- Пишем gpu-utilities
  - https://github.com/kolesa-team/gpu-utilities
- Используем NVML (NVIDIA Management Library)

kolesa group

Что я хотел бы знать об MLOps год назад

Message to my past self

- Single source of truth
- Модели без данных ничто
  - Сначала данные, потом ML

- Версионирование: данные + модель + код
- Панацеи нет: писать код придётся
  - MLOps инженер — в первую очередь разработчик

- Железо важно
- Мониторинг важен

kolesa group

Миржан Иркегулов


irkegulov@kolesa.kz


@sxingxi

kolesa group

Спасибо! Вопросы?