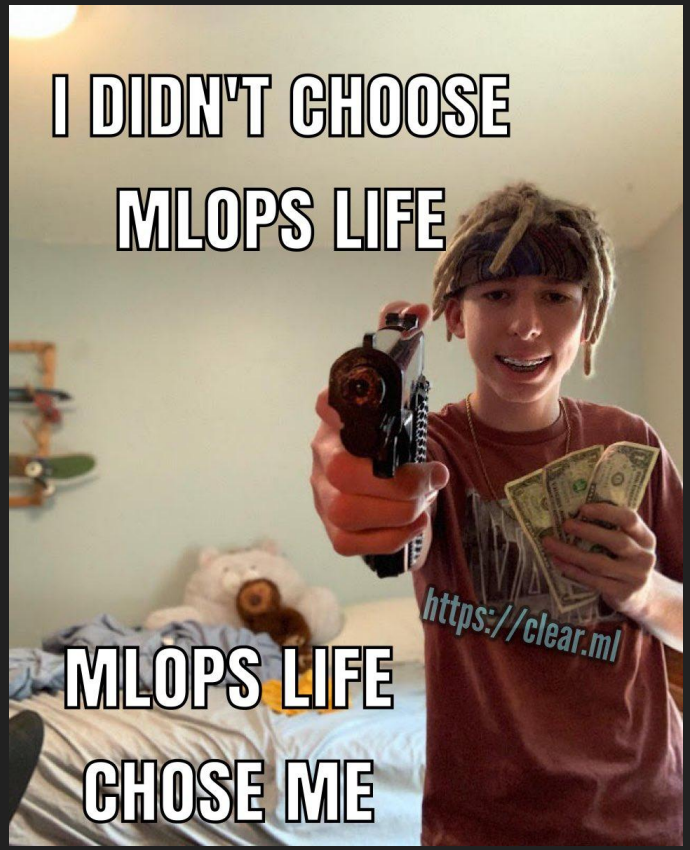
Цикл жизни ML-моделей в Cherry Labs

Кирилл Рыбачук, ML Engineer / Researcher @ Cherry Labs

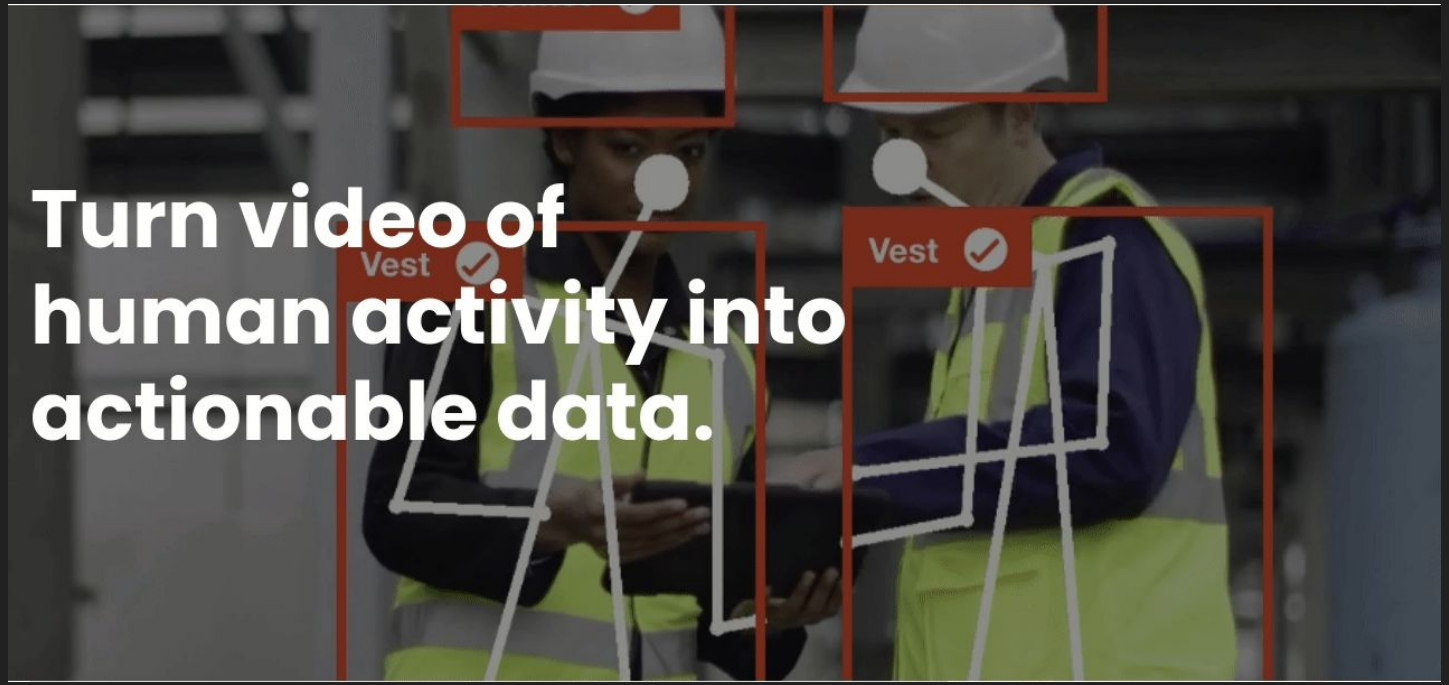
Немного обо мне

- MIPT’2013
- 8 лет в ML, 5 лет в CV, 4 года в CherryLabs
- Мамкин млопсер



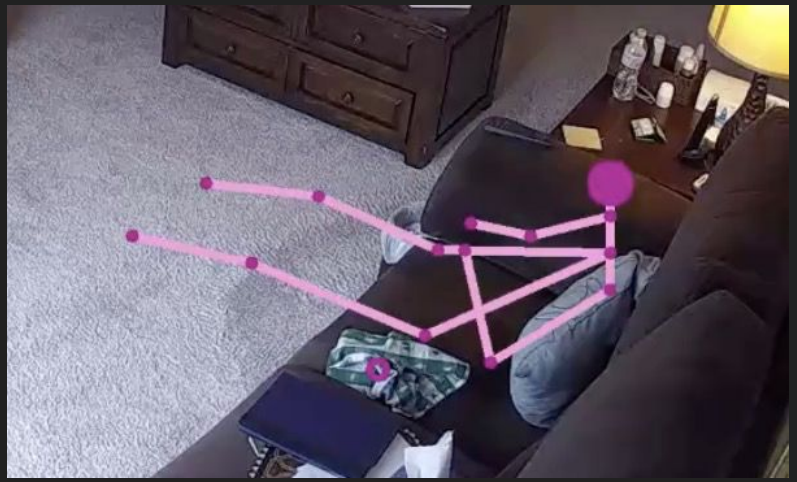
Что мы делаем

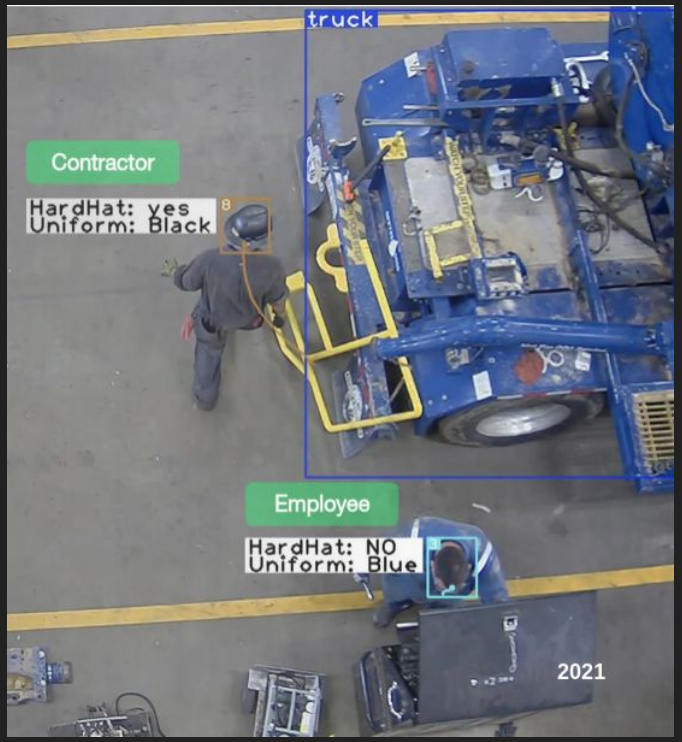
- Распознаем действия по видеопотоку
- Safety enforcement, efficiency control
- Cherry Home / Cherry Pro / Cherry Welcome



Примеры кейсов

- PPE Compliance (каски, униформы, обвязки...)
- Измерение температуры при входе в помещение
- Детекция падений
- Статистика по действиям на стройплощадке
- Progress bar производственного процесса

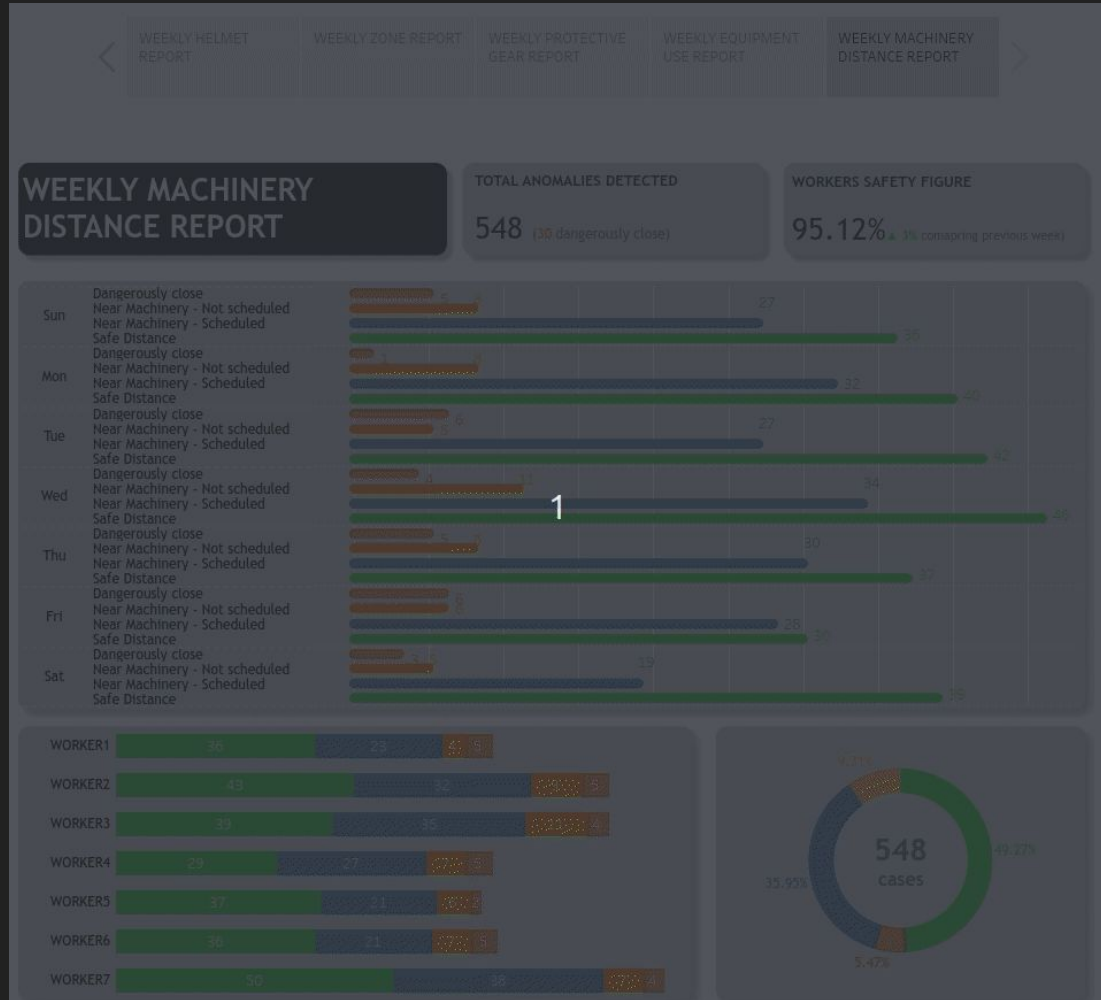




Что на выходе

Реалтаймовый фид событий

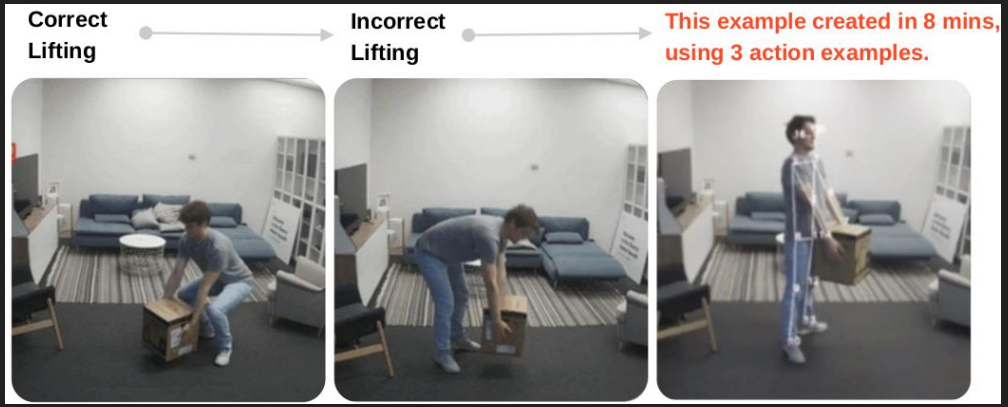
Оффлайновая аналитика за период



Что на выходе

Computer Vision tasks

- Object detection & Tracking
- Multi-person pose estimation
- Person re-identification
- Anomaly detection
- Action recognition (+ few-shot)
- Semantic segmentation

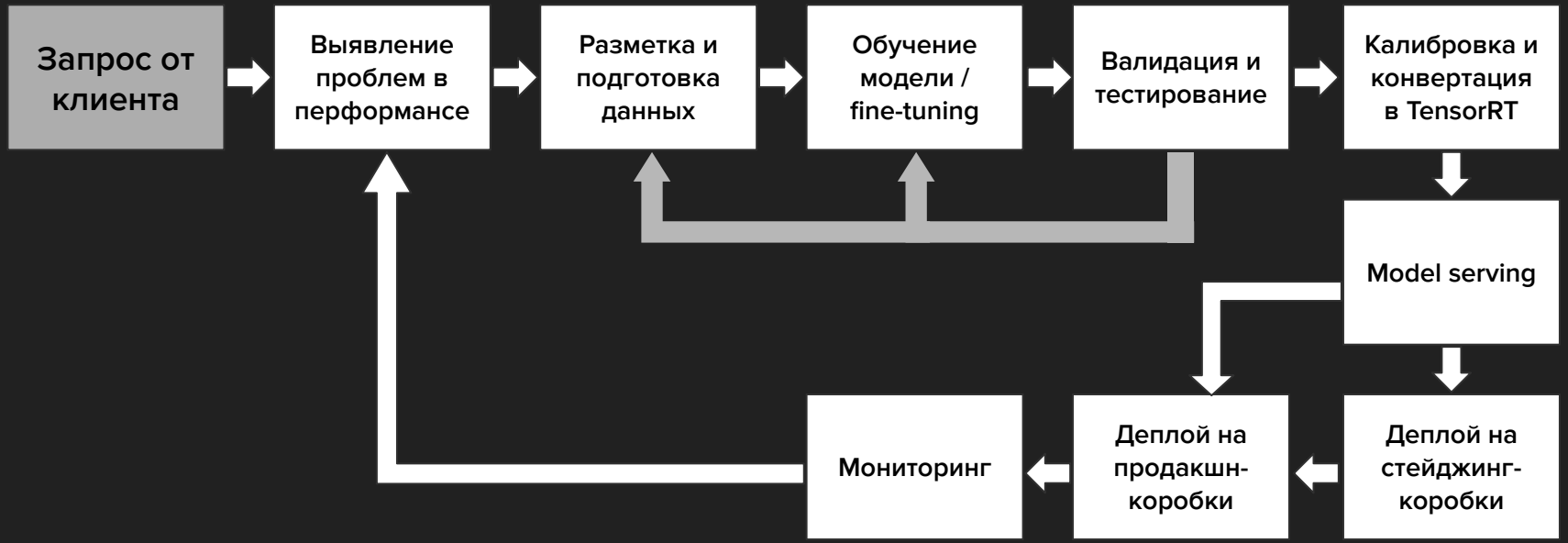


Зачем столько моделей

- Своя комбинация (протокол) для каждого клиента
- Нужно всем: детекция или оценка поз
- Нужно не всем: все остальное поверх детекции
- Много эвристик поверх ML-моделей (пространственная логика)
- Последовательное выполнение => ошибка стакается

От детекции зависит все остальное => надо мониторить и обновлять

Типичный пайплайн (цикл жизни) ML-модели



Ограничения & Что учесть

- Неопределенность требований?
  - Насколько актуальна будет эта задача в будущем
- Как часто переобучать?
- Сколько ресурсов на автоматизацию?
- Кто будет этим заниматься?
  - DS и DE - одни и те же люди
  - 1 ML Engineer и 1 DevOps - должно хватить
- Trade-off автоматизация vs гибкость экспериментов
  - Пайплайны нужны и для ресерча => нужно и то и то

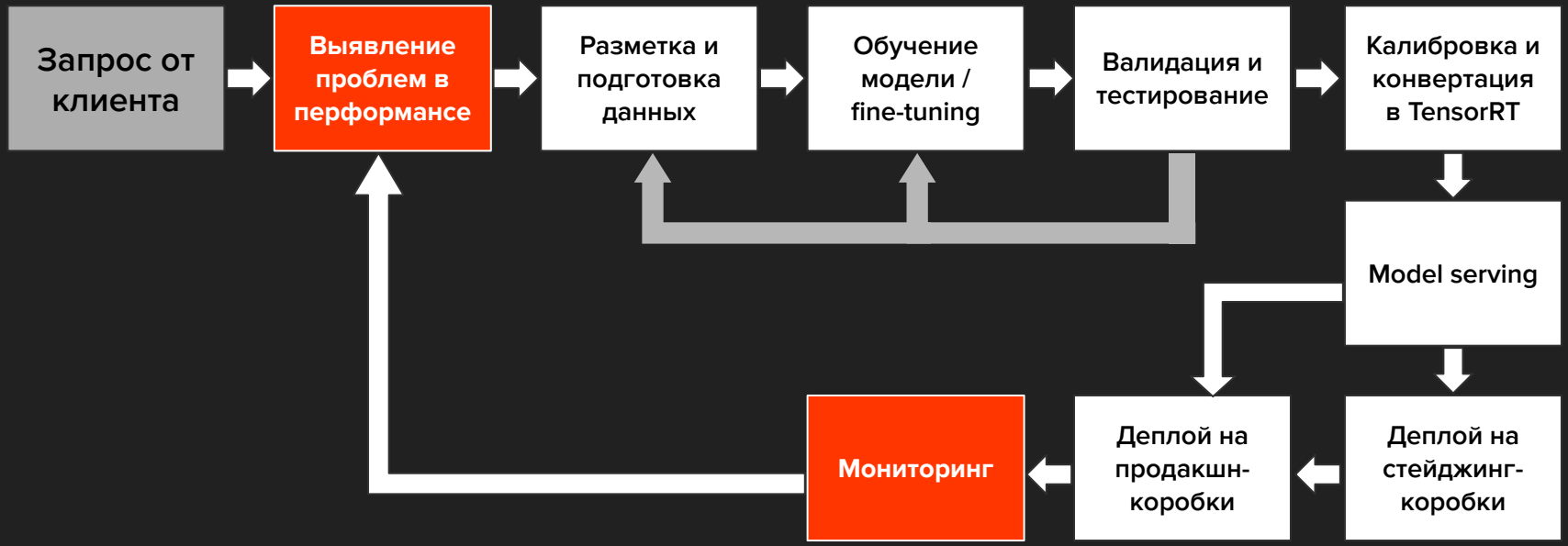
Особенности MLOps для Computer Vision

Сложно:

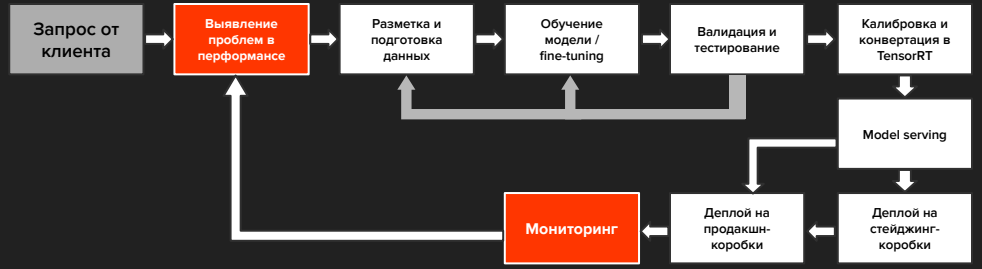
- Нет онлайнового фидбека от пользователей
- Редко возможна полностью автоматическая разметка
- Почти невозможны A/B тесты
- Почти невозможно автоматически детектить data drift
- human-in-the-loop для мониторинга и для разметки

Просто:

- Проще тестировать и версионировать данные
- Не нужен feature store (нет ручного инжиниринга фич)
- Нет training-inference skew
- Нет concept drift

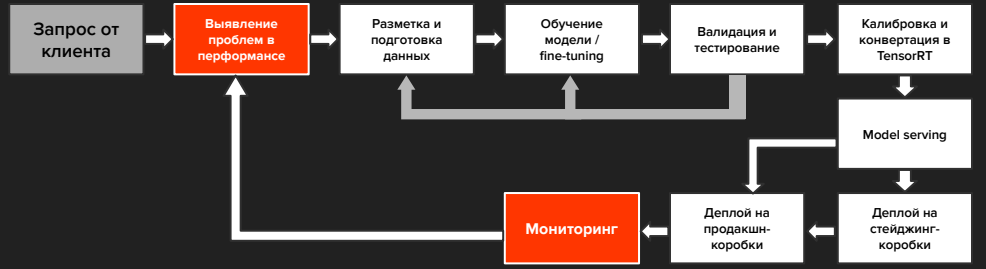

Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг

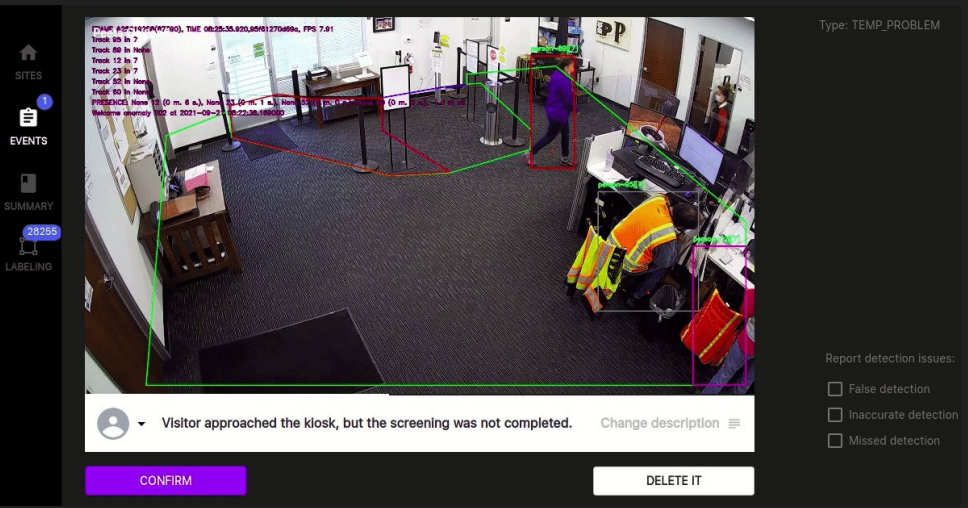
Human-in-the-loop

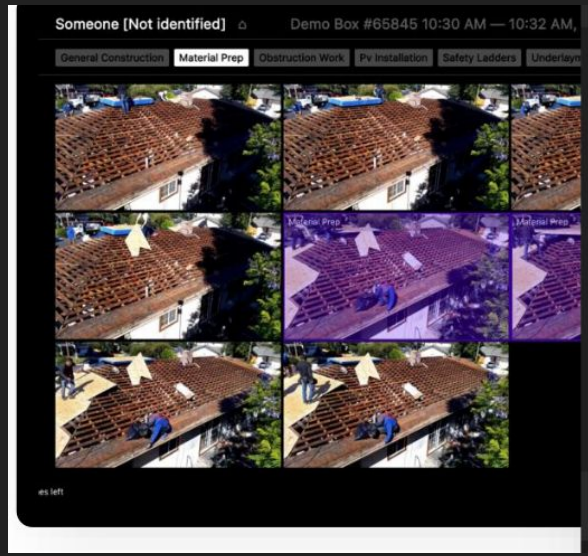


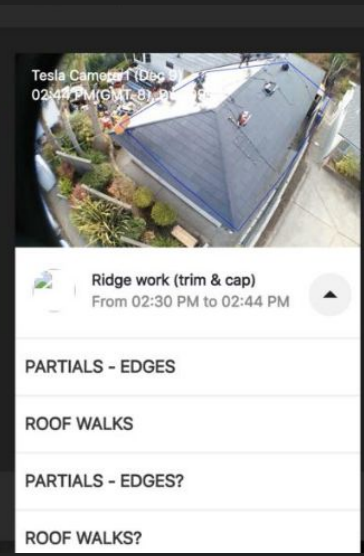
- Цена ошибки высока (и FN, и FP)
- Операторы в штате - очевидный выход
- Ручная проверка фида событий / аномалий => борьба с FP
- Ручная проверка разреженных кадров => борьба с FN

Human-in-the-loop

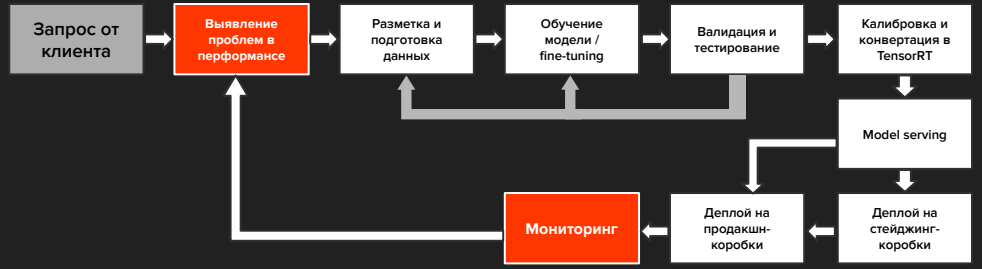






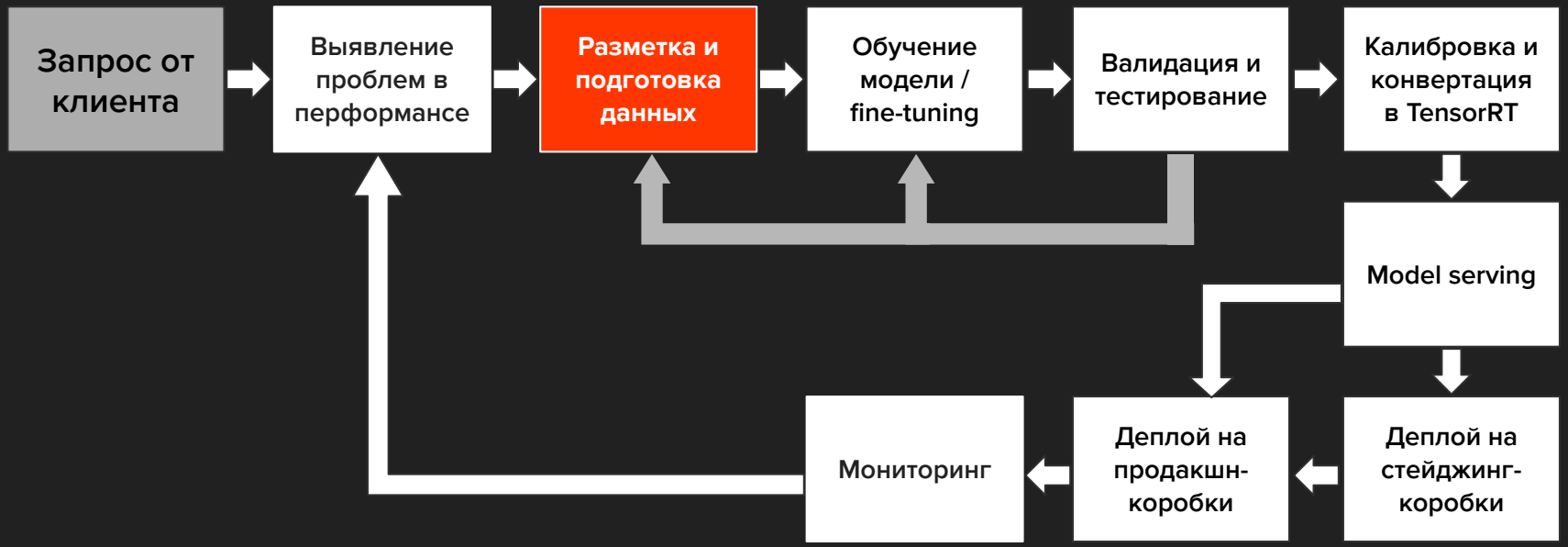


Human-in-the-loop



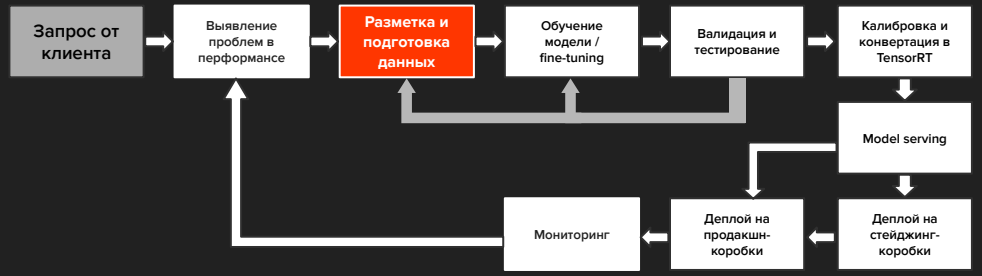
Side effects

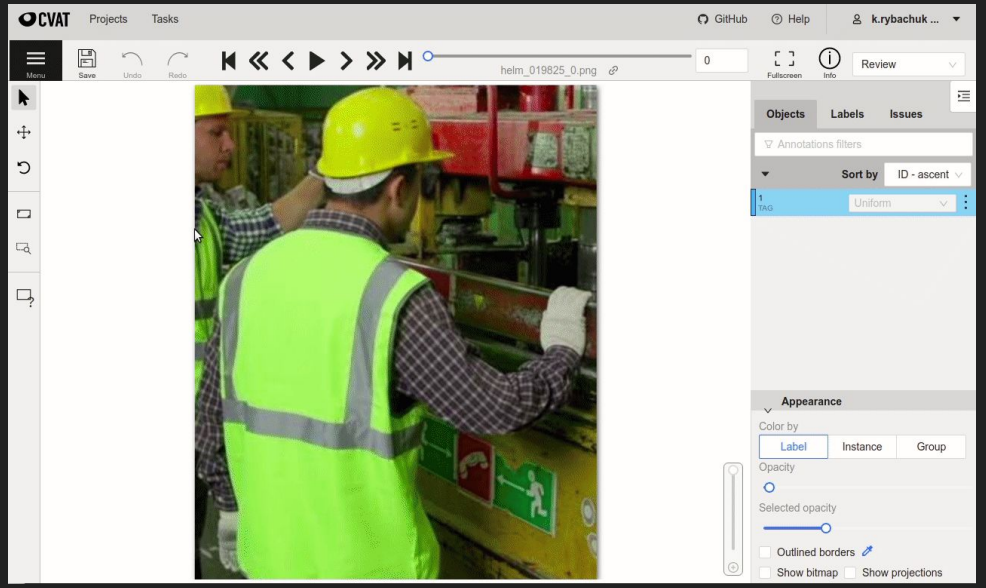
- Мониторинг
  - Операторы - основной способ делать это
  - Статистика по ложным срабатываниям / пропускам идет в дашборды
- Данные для дообучения
  - Репорты по сложным корнер-кейсам - операторы
  - Сырые данные для следующих версий датасетов - тоже операторы

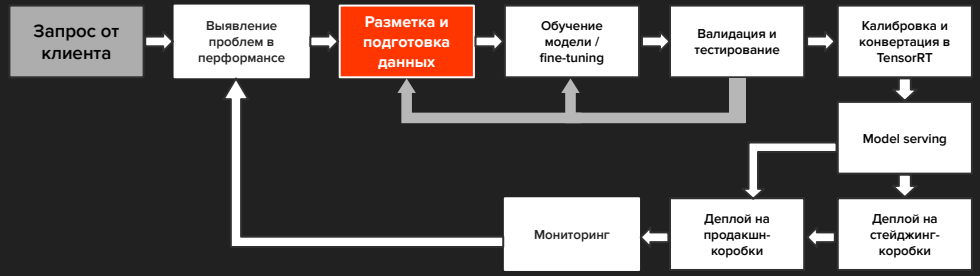

Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг

CVAT

- Много сценариев аннотации
- Внимание к деталям
  - Много хоткеев
  - Можно приближать
  - Grid lines для разметки bbox-ов
- Встроенный трекинг
- Можно делать псевдолейблы
- Много встроенных форматов I/O







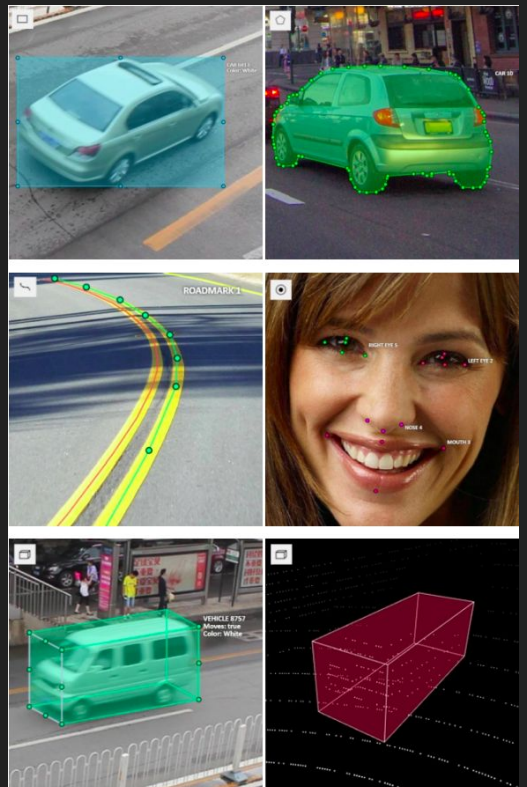
CVAT

Когда он хорош?

- Сложные лейблы
  - Сегментация, ключевые точки, 3D-детекция
- Разметка простых треков на видео
- Разовые задачи без версионирования
  - Разметка больших открытых датасетов
  - Напарсить и разметить

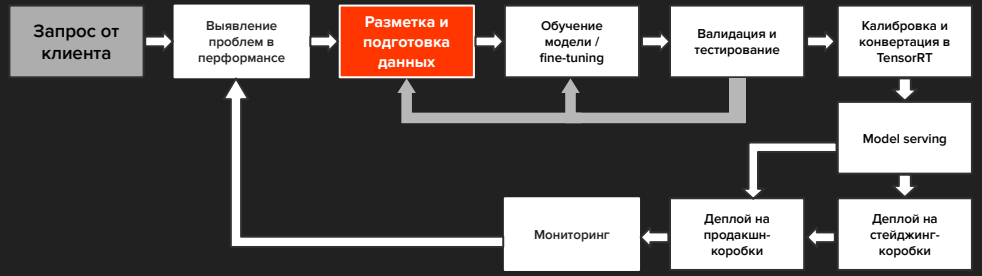
Когда он плох?

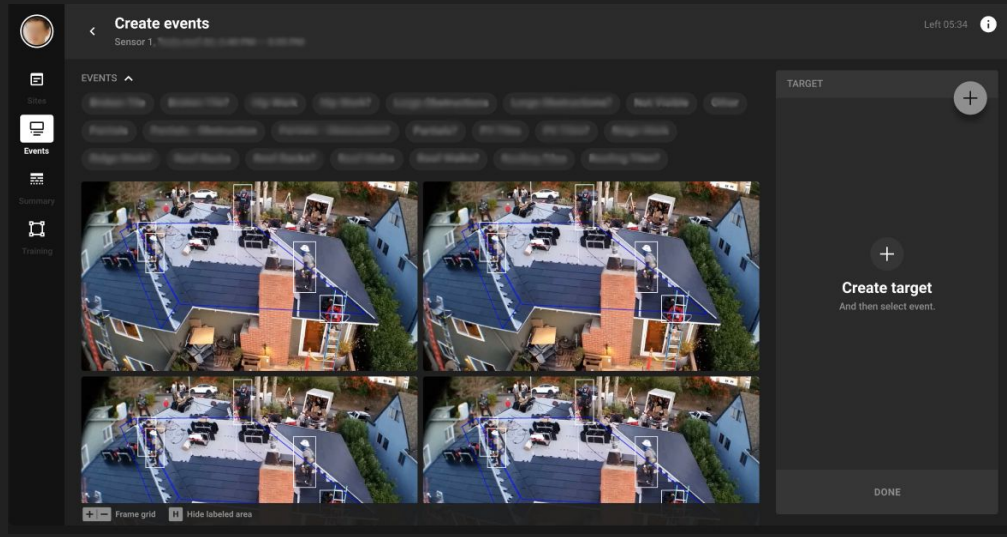
- Разметка сложных треков на видео
- Разметка произвольного диапазона кадров (например, действий)
- Быстрая разметка (он тормозной)
- Одно видео = одна джоба

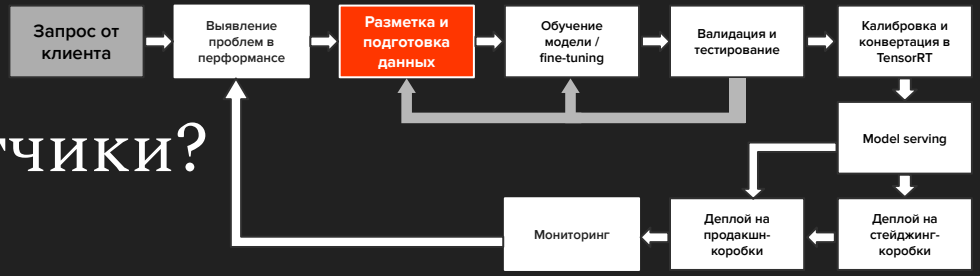


Внутренняя тулза

- Интегрирована в операторскую панель
- Могут размечать те же люди, кто репортил
- Удобно версионировать
  - Входная точка для ML-пайплайна
  - Результат сохраняется в S3
- Удобно размечать действия по серии кадров
- Не тормозит
- Псевдолейблы на вход

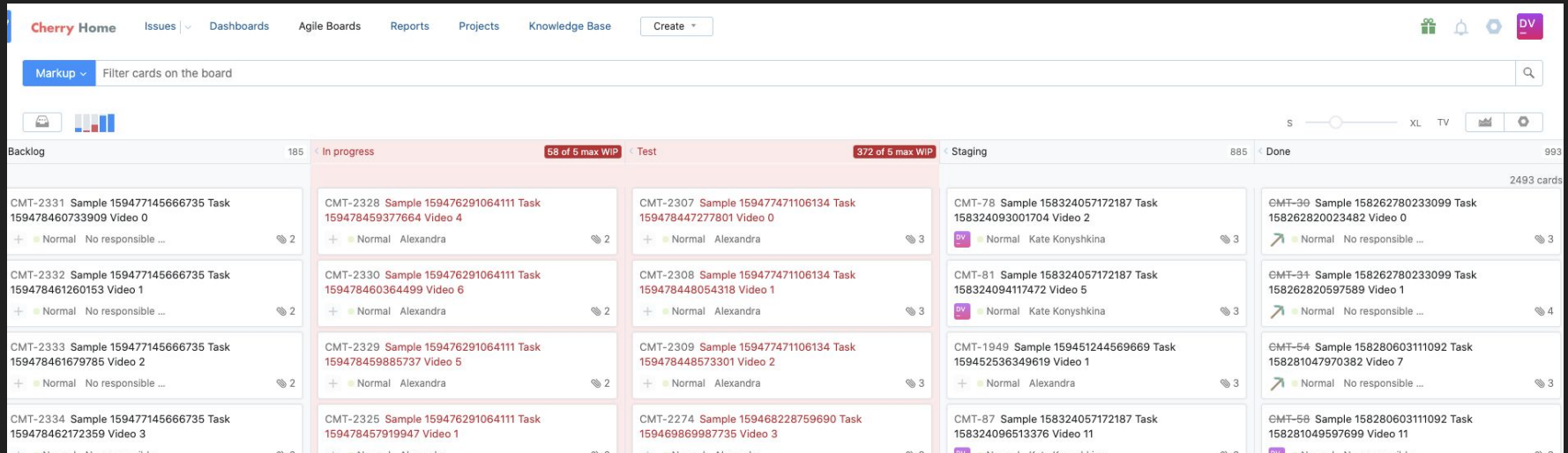


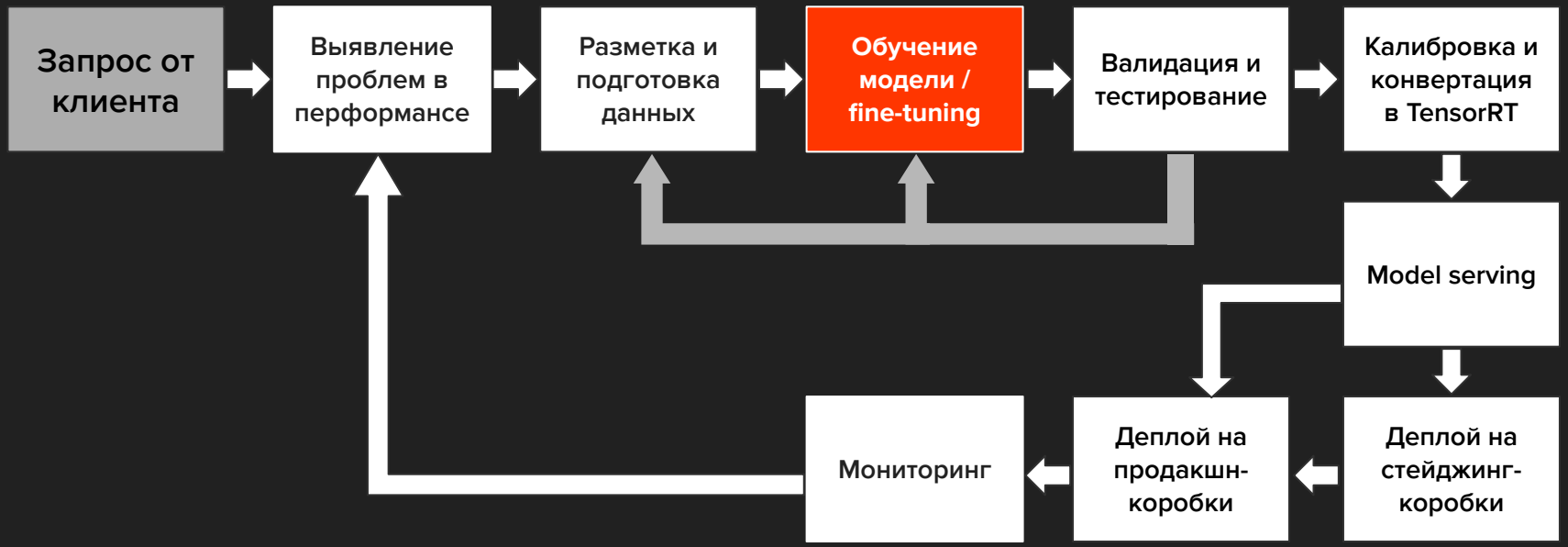


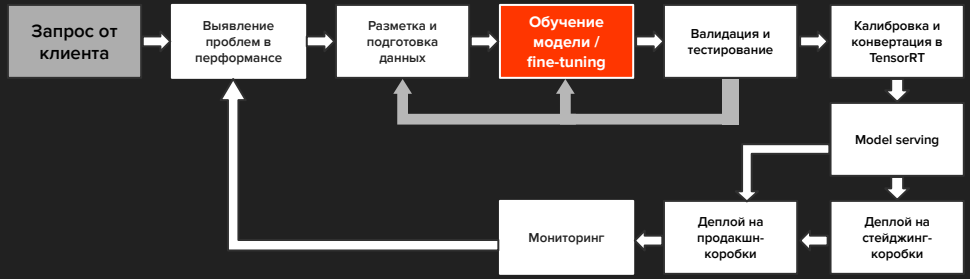


Что если внешние разметчики?

- Репорты автоматически создают задачи в Youtrack
- Выборочная проверка ✅
- Дублирование ❌
- Регулярный сбор задач в Done через Jenkins




Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг



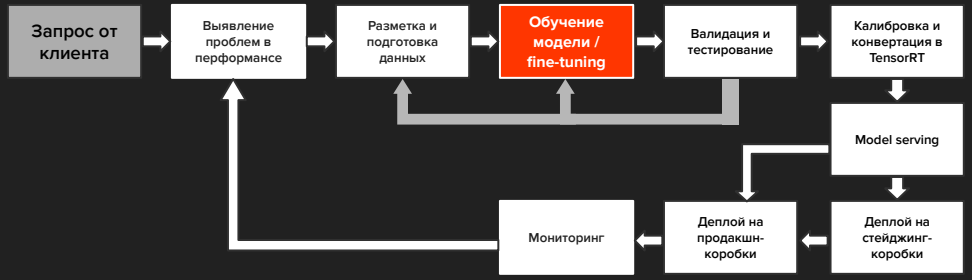
Эксперименты

Кейс CherryHome

- Много разнообразных indoor-сцен + корнер-кейсы
- Data drift
  - Освещение, новые вещи, перестановка мебели
- Редкость / непредсказуемость аномалий

=> Одна модель на всё с высокой обобщающей способностью

=> cron-команды через Jenkins для всех этапов пайплайна



Эксперименты

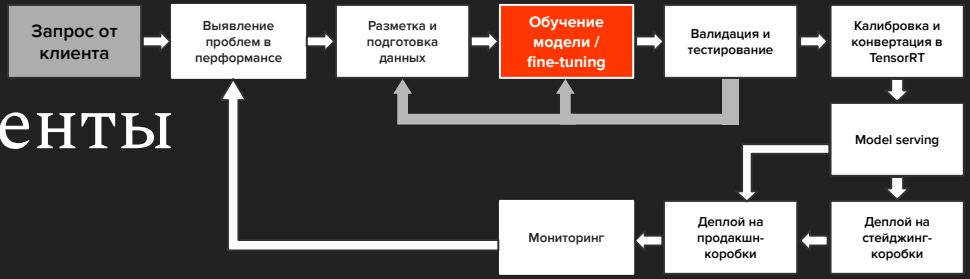
Кейс CherryPro

- Разные требования для разных клиентов
- Масштабируемость для старых клиентов
- Быстрые итерации для новых клиентов

=> Camera-specific models

=> Легко запутаться

Что делать?



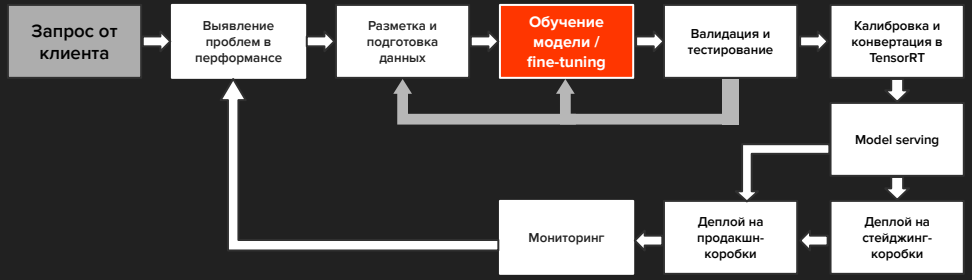
Как привести эксперименты в порядок

- Версионирование данных и моделей
- Воспроизводимость эксперимента и окружения
- Трекинг и сравнение экспериментов

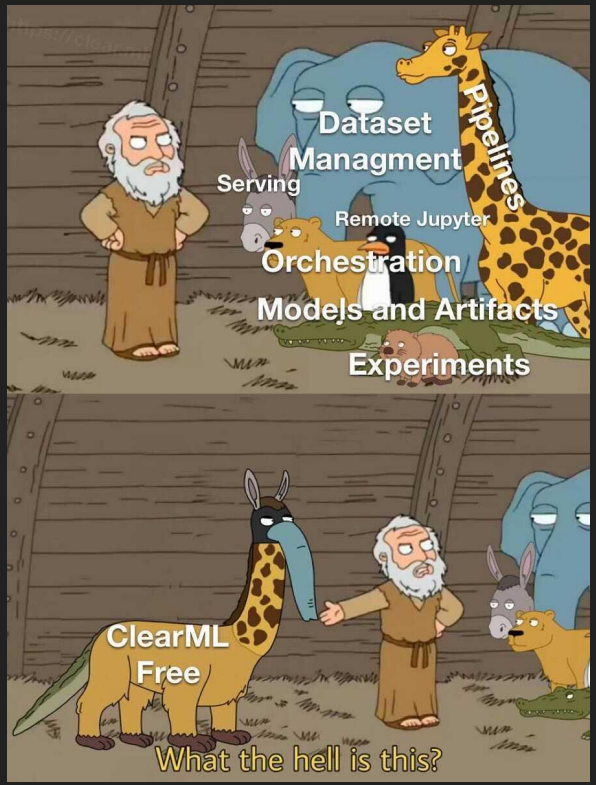
neptune.ai
polyaxon
comet
CLEAR | ML
mlflow
Weights & Biases
DVC + CML
guild.ai

YOU ARE NOT YOUR STACK

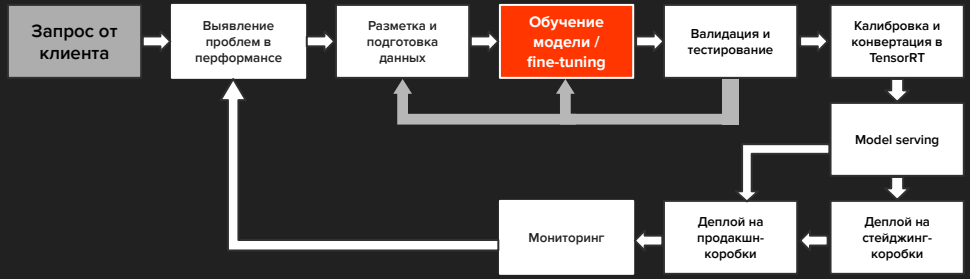
Сравнение MLOps-инструментов на сайтах этих инструментов be like



- Полностью open source
- clearml-server + clearml-agent + SDK (python package)
- Запуск экспериментов в очередях на агентах
- Контейнеризация всего
- Пытается стать всем сразу:
  - Эксперименты (версионирование, мониторинг, воспроизводимость, resourse management)
  - DataOps (dataset storage, version control)
  - Оркестрация (DAGs, scheduler)
  - CI/CD + serving
- Оживленный гитхаб и слек
  - github.com/allegroai/clearml
  - clearml.slack.com



Компоненты



clearml-server

- На EC2 / GCP / k8s / bare metal
- Web UI (app.*.*) + RESTful API (api.*.*) + file server (files.*.*)

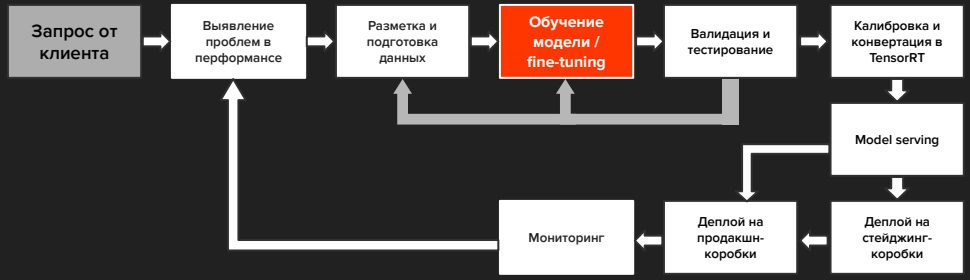
clearml-agent

- Несколько очередей на одном агенте:

```
clearml-agent daemon --detached --queue high_priority low_priority --docker
```

- Распределение GPU по очередям - платная фича

Откуда берутся таски




```python
import yaml
from clearml import Task

def get_args(cfg_path: str):
    cfg = yaml.load(open(cfg_path).read())

    task = Task.init(project_name='MyProject',
                     task_name='MyTask')
    for name in cfg:
        task.connect(cfg[name], name=name)

    # optionally
    task.execute_remotely(queue_name='my_queue')

    return task, cfg
```

ИЛИ

```python
source_task = Task.get_task(project_name='MyProject',
                            task_name='SourceTaskName')
task = Task.clone(source_task.id)
```

`Task.create` s broken! В версии 1.1.1 лучше не использовать

Все параметры таски можно менять и из кода, и из Web UI

RECENT PROJECTS

VIEW ALL

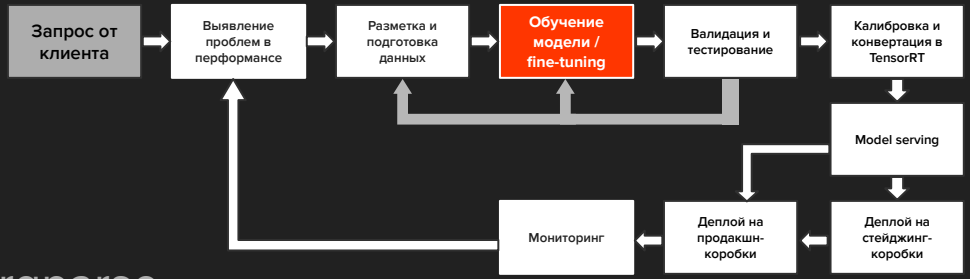
+ NEW PROJECT

docs_examples	examples (1 sub projects)	Cifar	DevOps
TOTAL: 10	TOTAL: 42	TOTAL: 3	TOTAL: 1
RUNNING: 1	RUNNING: 0	RUNNING: 0	RUNNING: 1
COMPLETED (24 hrs): 0	COMPLETED (24 hrs): 0	COMPLETED (24 hrs): 0	COMPLETED (24 hrs): 0
COMPUTE TIME: 00:34:44	COMPUTE TIME: 10:35:20	COMPUTE TIME: 00:00:00	COMPUTE TIME: 00:00:00

RECENT EXPERIMENTS

MANAGE WORKERS AND QUEUES

TYPE	TITLE	PROJECT	STARTED	UPDATED	STATUS
Training	Keras with TensorBoard example	docs_examples	Jun 8 2021 14:44	Jun 8 2021 14:52	Running
Training	tensorboard pr_curve	docs_examples	Jun 8 2021 14:42	Jun 8 2021 14:51	Completed
Training	tensorboard toy example	docs_examples	Jun 8 2021 14:41	Jun 8 2021 14:50	Completed
Training	pytorch with tensorboardX	docs_examples	Jun 8 2021 14:45	Jun 8 2021 14:49	Completed
Training	Tensorflow v2 mnist with summari...	docs_examples	Jun 8 2021 14:41	Jun 8 2021 14:46	Completed



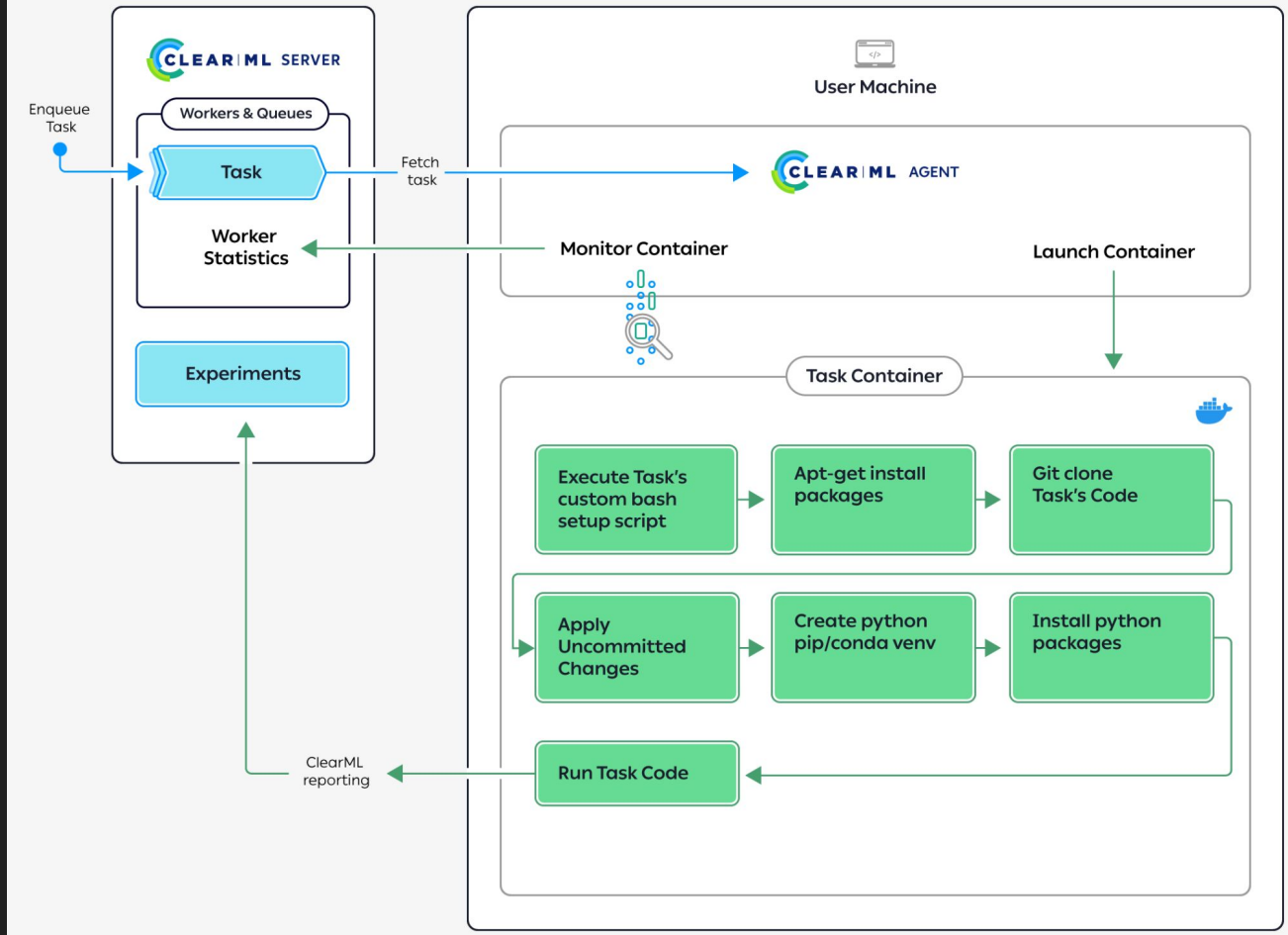
Лайфхак

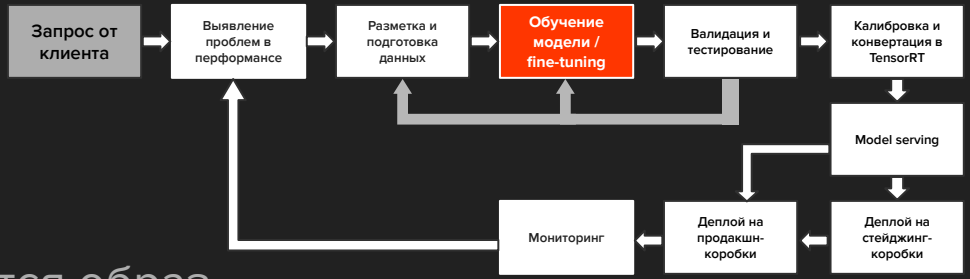
- Не коннектить параметры через Argparse

```
parser = ArgumentParser()
parser.add_argument('--awesome-arg',
                    help='i am awesome',
                    type=int, default=1)

args = parser.parse_args()

task = Task.init(...)
```

- Они имеют приоритет над вкладкой Configuration
- Что бы ни отображалось в Web UI, передаваться будут дефолтные значения

ClearML Agent flow diagram





Лайфхак

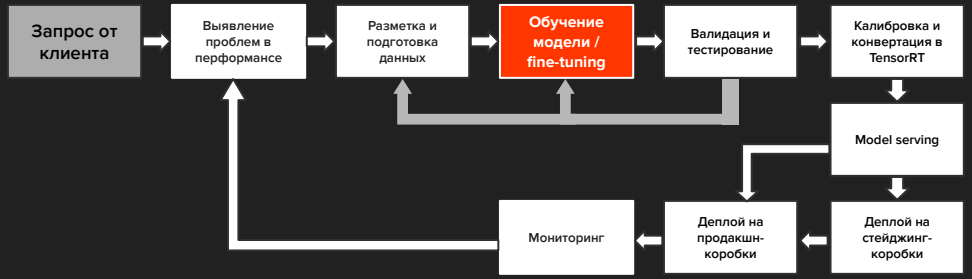
- При каждом запуске пересобирается образ
+ Можно не пересобирать образ самому при каждом коммите
- Зато огромный overhead! Что делать?

Решение:

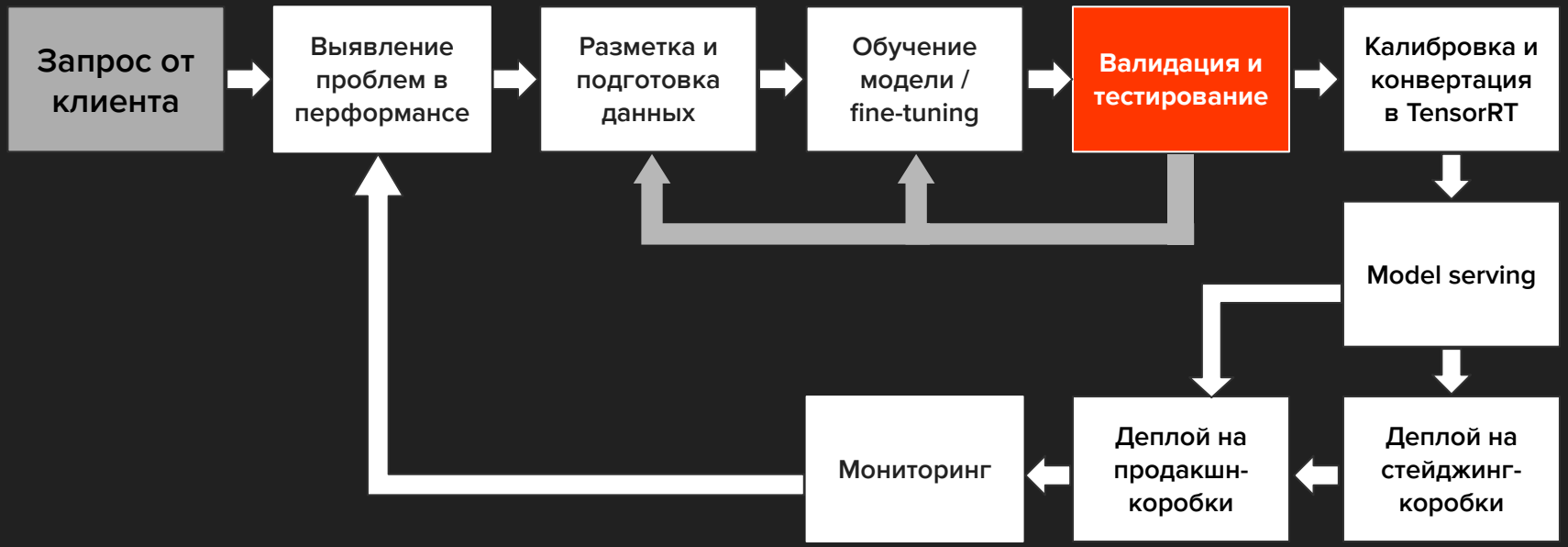
- Пустой requirements.txt
- Базовый (тяжелые слои) + запускаемый (легкие слои) образа

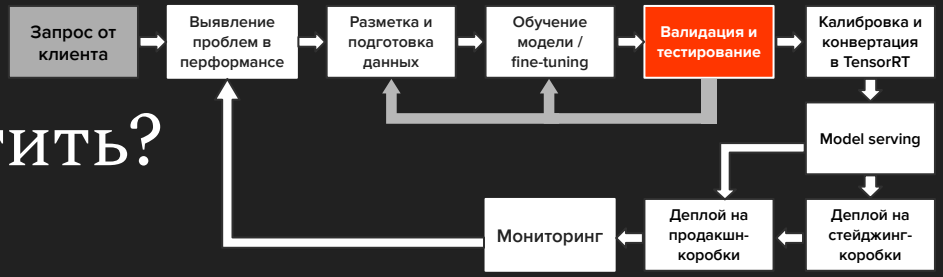
```
FROM registry.gitlab.com/cherrylabs/ml/clearml-demo:base

ENV workdir /code/workdir
WORKDIR ${workdir}

COPY util ${workdir}/util
COPY cfg ${workdir}/cfg
COPY *.py *.json ${workdir}/
```

Артефакты



- На вход: версионированные данные из S3
  - Список слайсов для обучения и валидации
- Синхронизация датасетов и исходных моделей с S3 на агента
  - Собственный велосипед
  - `-v /opt/clearml/data:/data`
- Сейчас бы сделал по-другому: clearml-data и clearml.Dataset
- На выход: сохранение лучших чекпойнтов в S3


Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг



Как понять, что пора катить?

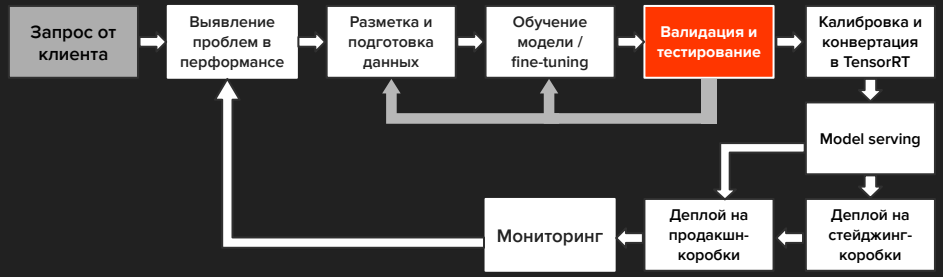
Регрессионные тесты

- Если есть стабильная задача со слабым data drift

Датасеты

- Открытые (COCO, OpenImages, kaggle datasets...)
- Вручную собранные корнер-кейсы
  - Реальные
  - Из открытых источников
  - Синтетические
- Версии наших внутренних датасетов

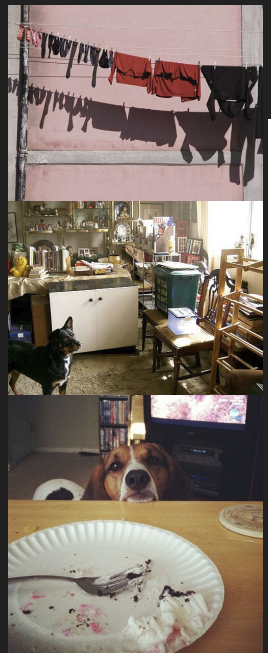
Новая модель на старых данных	Старая модель на старых данных
Новая модель на диффе	Старая модель на диффе

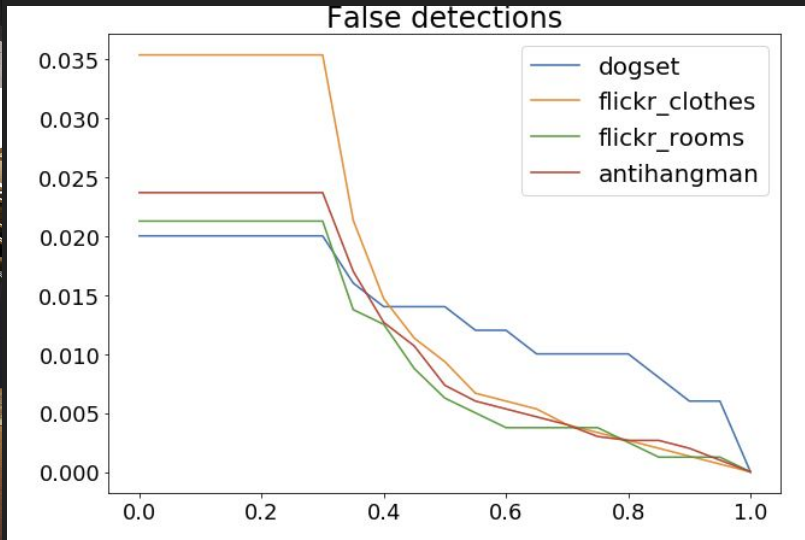


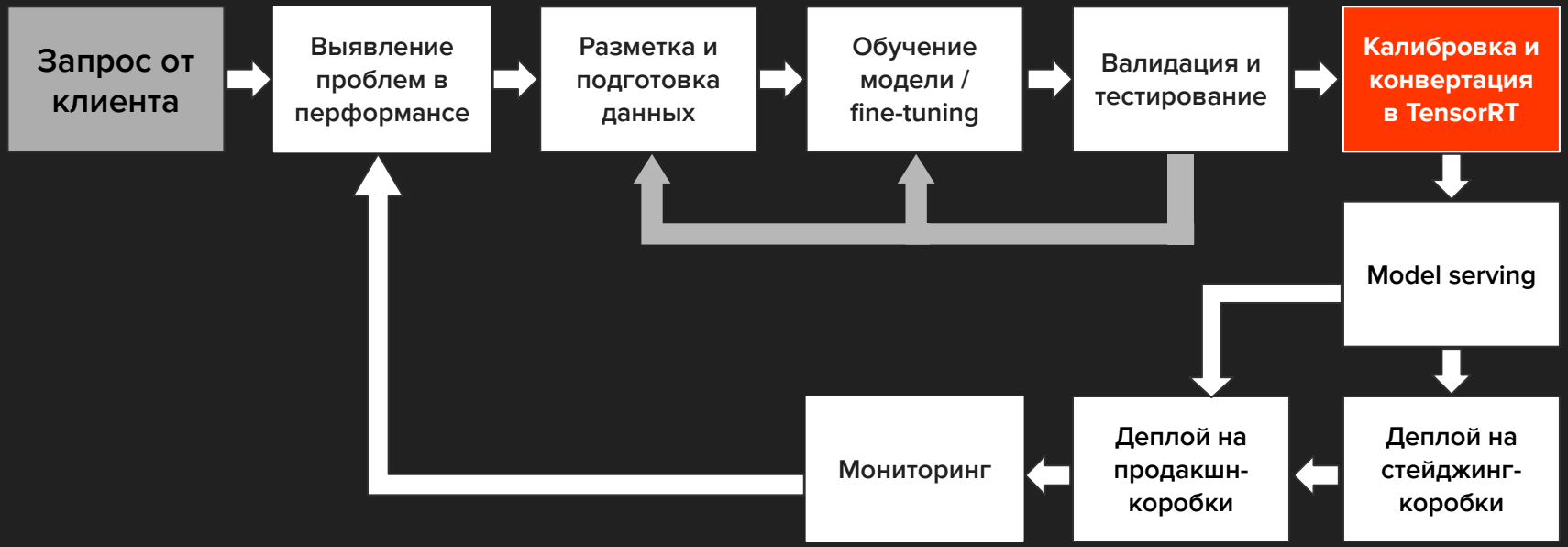
Корнер-кейсы

Пример CherryHome

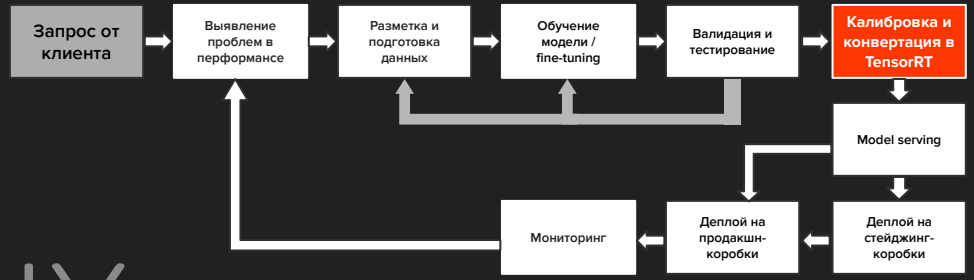
1. Сложные indoor-сцены без людей для борьбы с FP
   a. Достаем подходящие классы из открытых датасетов, фильтруем людей
   b. Парсим Flickr, фильтруем людей
   c. Собираем датасет собак
2. Тестирование на FN
   a. Unity-сгенерированные сцены
   b. Падения реальных бабушек






Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг

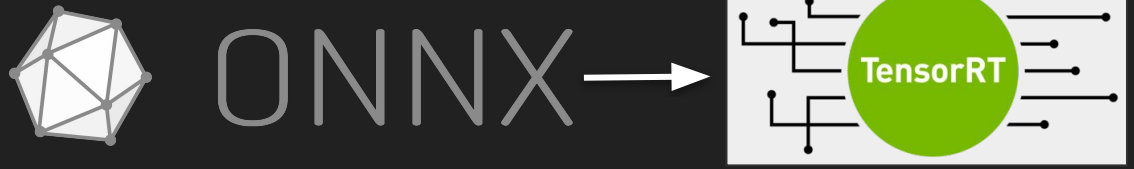
Конвертация

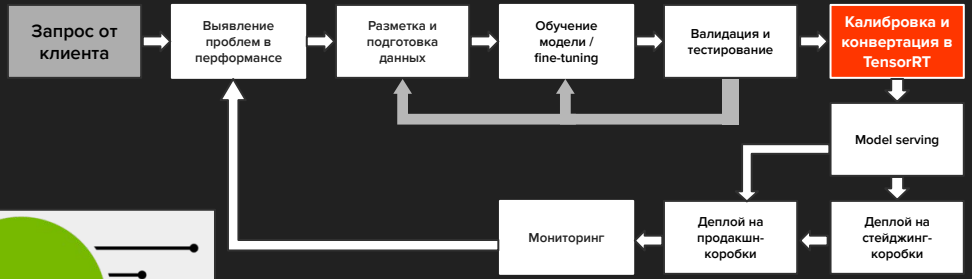


.pth → ONNX

```
input_names = ["input"]
output_names = ["boxes", "confs"]
dynamic_axes = {"input": {0: "batch_size"},
                "boxes": {0: "batch_size"},
                "confs": {0: "batch_size"}}
x = torch.randn((batch_size, 3, IN_IMAGE_H, IN_IMAGE_W), requires_grad=True)
onnx_file_name = "{}.onnx".format(OUT_NAME)
torch.onnx.export(model,
                  x,
                  onnx_file_name,
                  export_params=True,
                  opset_version=11,
                  do_constant_folding=True,
                  input_names=input_names, output_names=output_names,
                  dynamic_axes=dynamic_axes)
```

Конвертация

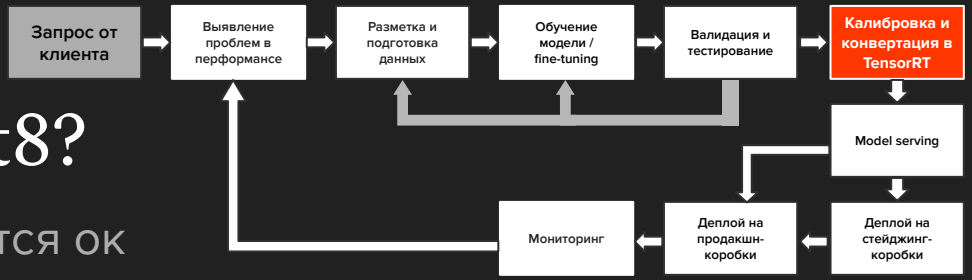





```
trtexec --onnx=/data/${basename}.onnx \
        --minShapes=input:1x3x640x640 \
        --optShapes=input:8x3x640x640 \
        --maxShapes=input:8x3x640x640 \
        --workspace=2048 \
        --saveEngine=/data/${basename}.engine
```

- Собирать надо на той же compute capability, на которой будем применять!
- Много видов GPU в продакшне => используем несколько агентов

```
from py3nvml.py3nvml import nvmlInit, nvmlDeviceGetHandleByIndex, nvmlDeviceGetName
nvmlInit()
gpu_obj = nvmlDeviceGetHandleByIndex(0) # 0th device
device_name = nvmlDeviceGetName(gpu_obj)
```



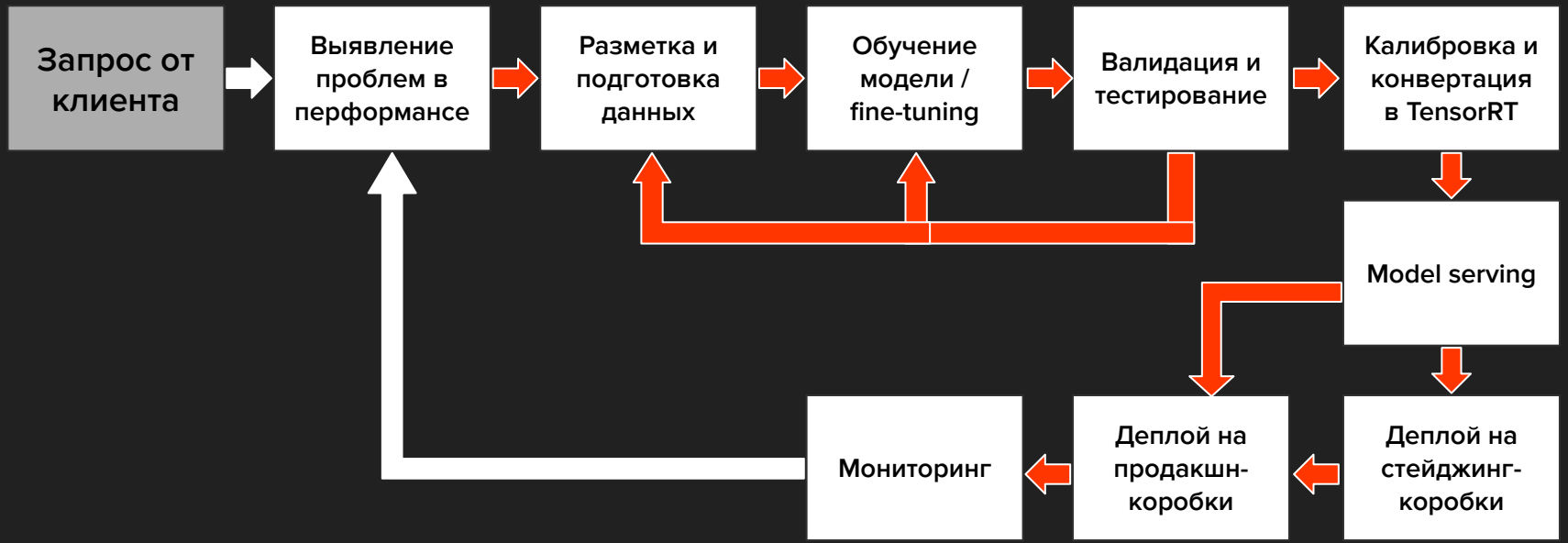
Конвертировать ли в int8?

- Некоторые модели - конвертируются ок
- Другие - проседает качество

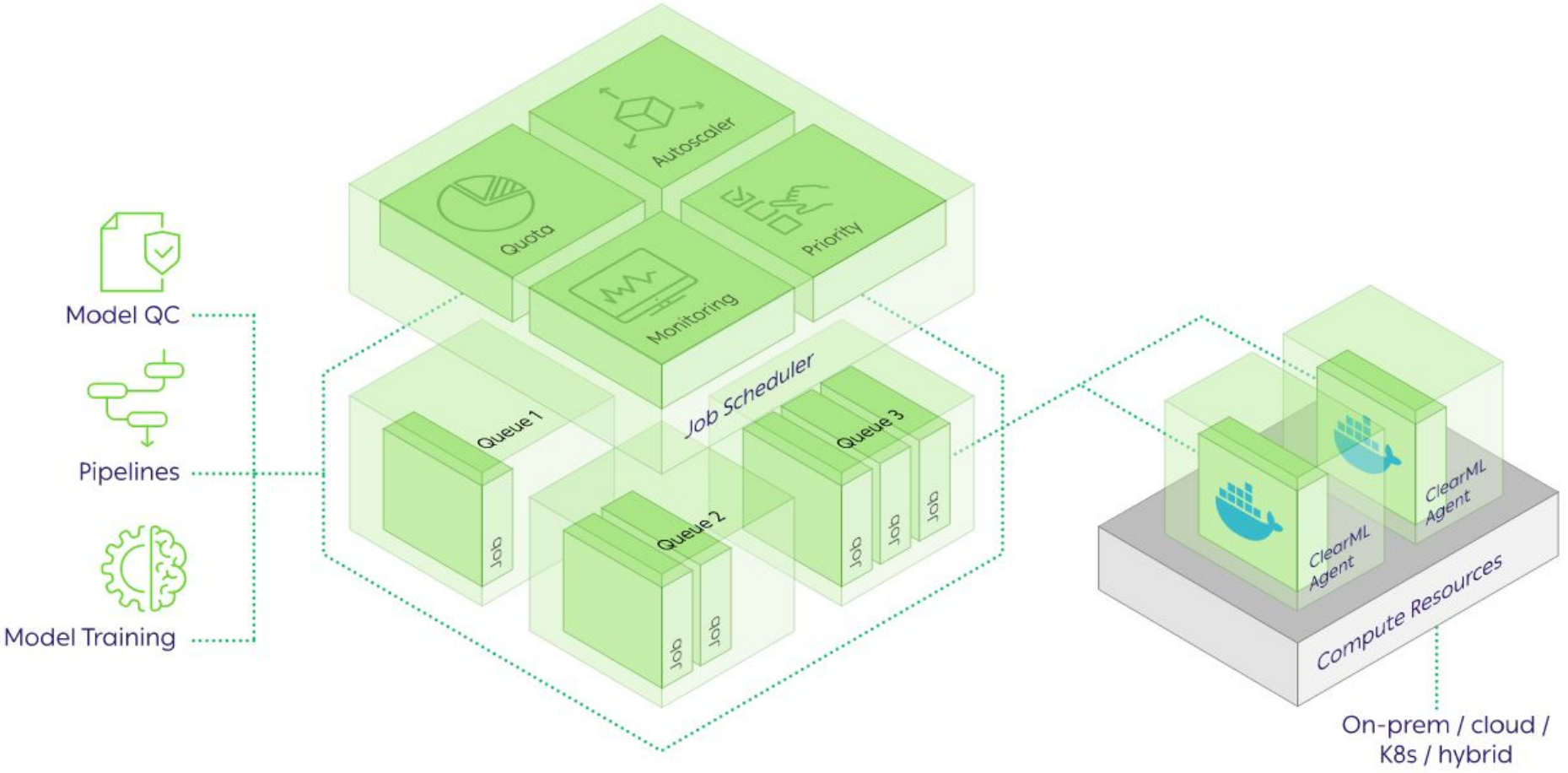
От чего зависит?

- Калибровочные данные должны быть:
  - Похожи на данные в проде
  - Не слишком далеко от данных в трейне
  - Больше != Лучше
- Разный масштаб весов в разных каналах в слое
- `torch.nn.qat` иногда помогает

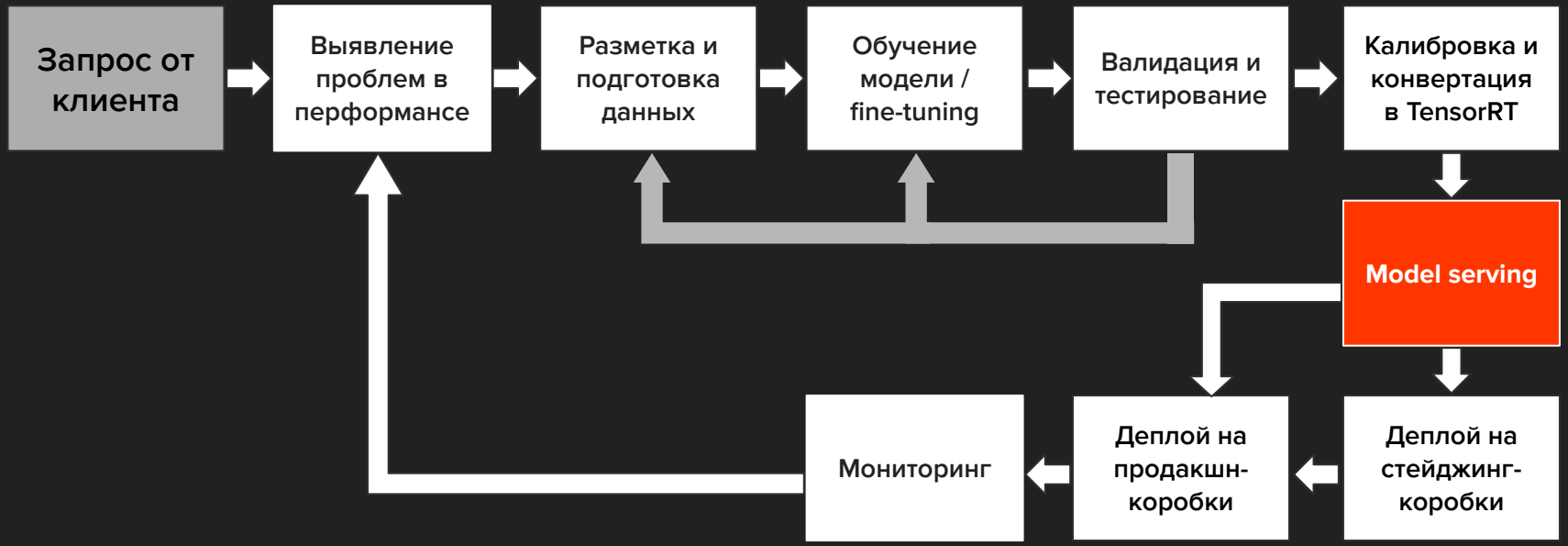
Оркестрация



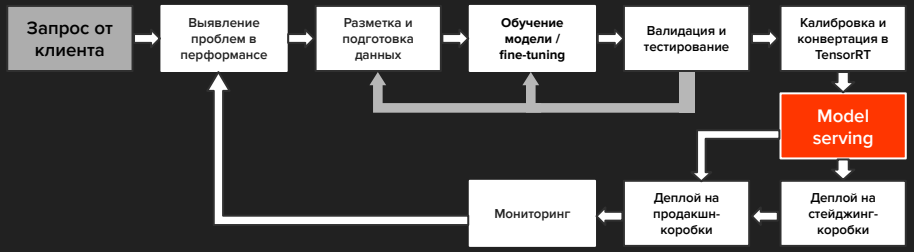
Orchestration with ClearML




my demo code
BATTLE-TESTED
OPEN-SOURCE
MLOPS ENGINE
ClearML


Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг

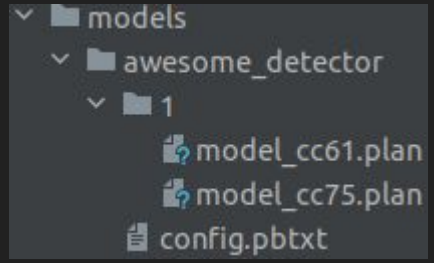
NVIDIA
TRITON INFERENCE SERVER



- Запросы через HTTP или gRPC
- Удобная структура хранения
- Встроенное версионирование
- version_policy: all, specific, latest

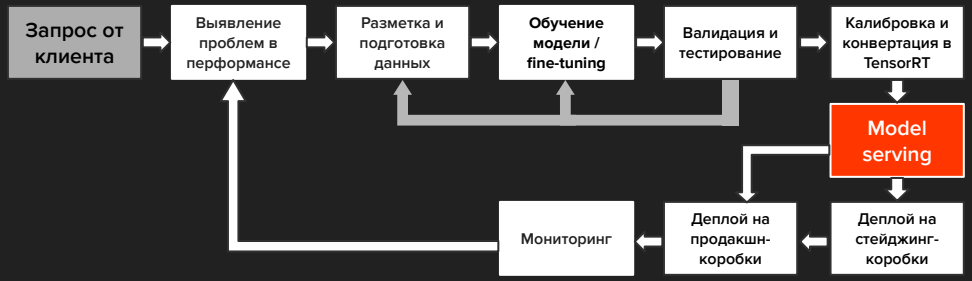
```
version_policy: { all { }}
```

```
version_policy: { latest { num_versions: 1 }}
```




```
name: "awesome_detector"
platform: "tensorrt_plan"
version_policy {
  latest {
    num_versions: 1
  }
}
max_batch_size: 8
input {
  name: "input"
  data_type: TYPE_FP32
  dims: ...
}
output [
    ...
]
cc_model_filenames [
  {
    key: "7.5"
    value: "model_cc75.plan"
  },
  ...
]
```

- Проблема 1: ВСЕ версии одной модели грузятся в GPU
- Решение: использовать только одну последнюю
- Версионирование своё, через имена моделей



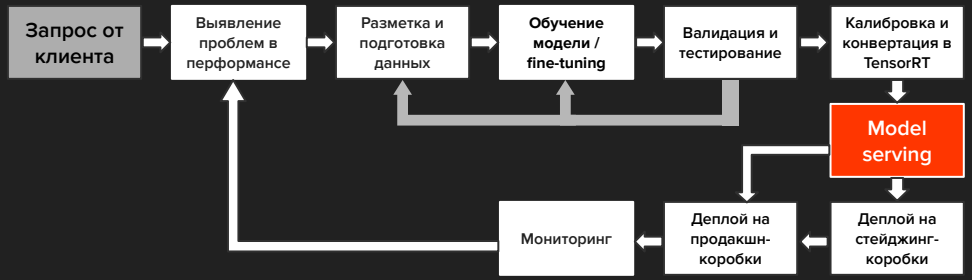
Где хранить модели?

- В репозитории? => супер-тяжелый образ и репа
- В S3? => нет нормальной встроенной синхронизации
  - При каждом запуске контейнера нужно качать заново!

Что делать?

- Рукописный синк моделей
- Триггер - любое изменение конфигов

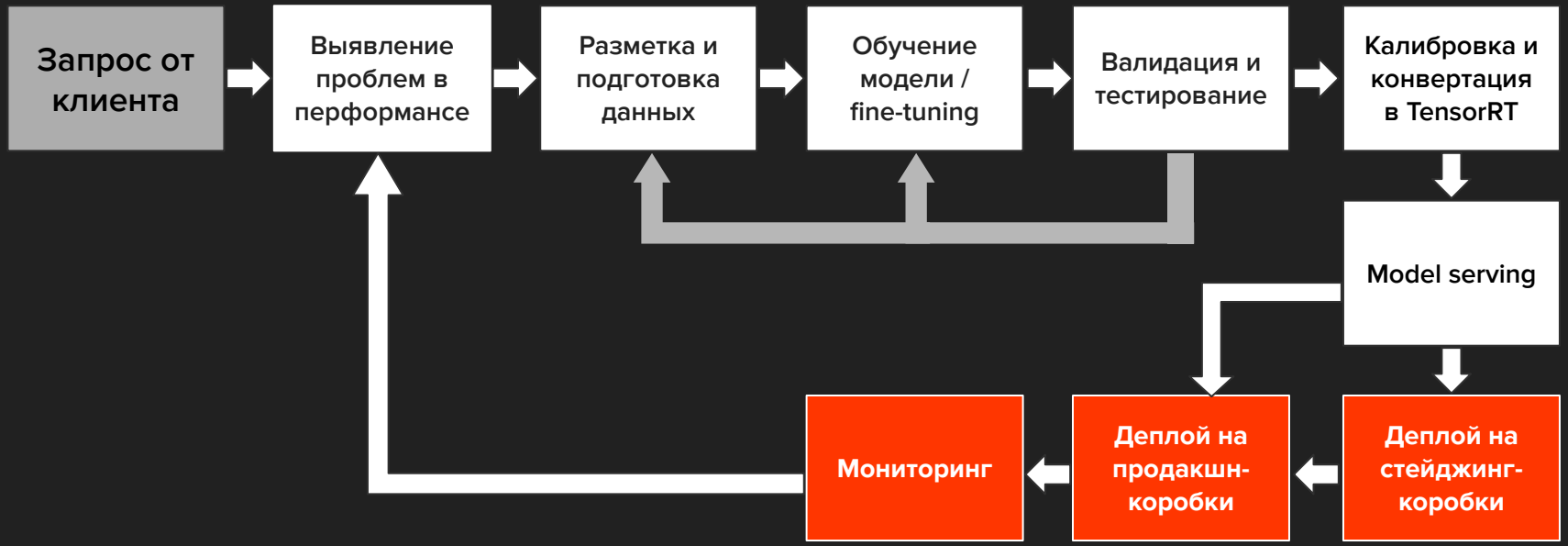
NVIDIA
TRITON INFERENCE SERVER



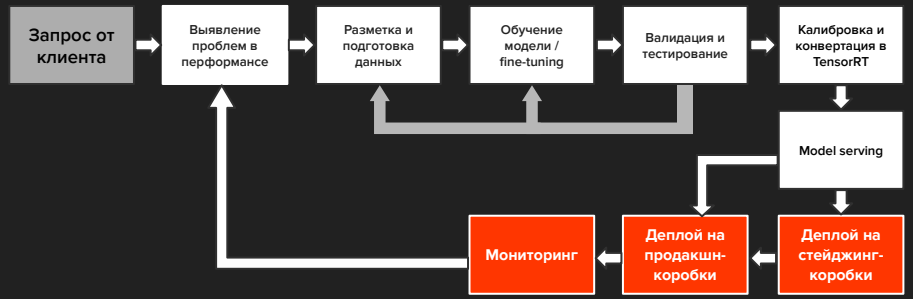
- Минус: неинформативные ошибки
- В любой ситуации

```
tritonclient.utils.InferenceServerException: no version is available
```

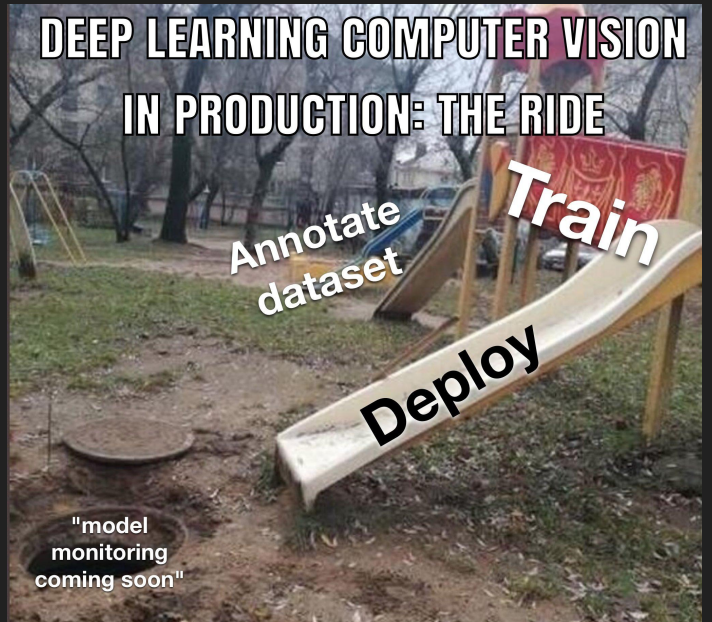
- Что делать: терпеть


Запрос от клиента
Выявление проблем в перформансе
Разметка и подготовка данных
Обучение модели / fine-tuning
Валидация и тестирование
Калибровка и конвертация в TensorRT
Model serving
Деплой на стейджинг-коробки
Деплой на продакшн-коробки
Мониторинг

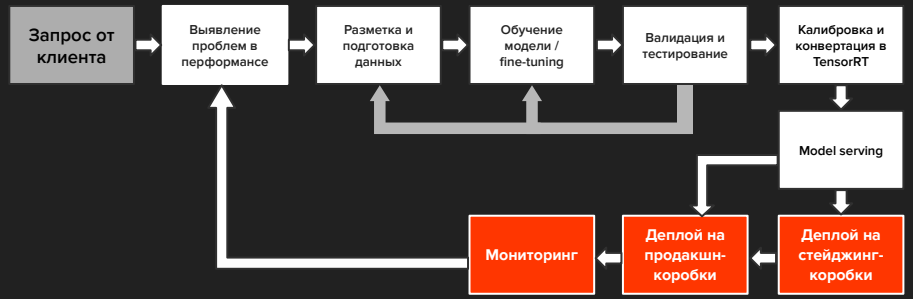
Деплой



- Тестовые коробки: в помещениях с QA-командой
- Стейджинг: кто согласится быть испытательным полигоном (не на каждом проекте)
- Мониторинг
  - Grafana - технические метрики
  - Операторы (human-in-the-loop) - все остальное



Результат



- Сокращение времени разработки на одну итерацию
  - Было - 4-7 дней
  - Стало - 10-15 минут
- Вовлеченность всех DS, коллаборация, воспроизводимость

Что можно улучшить:

- clearml.Dataset вместо велосипедов
- Автоматическая конфигурация пайплайна
- Интерпретируемые критерии деплоя

Спасибо за внимание!

Contacts:
telegram: @kirillfish
instagram: @kirillrybachuk
ods: #kurtosis

We are hiring