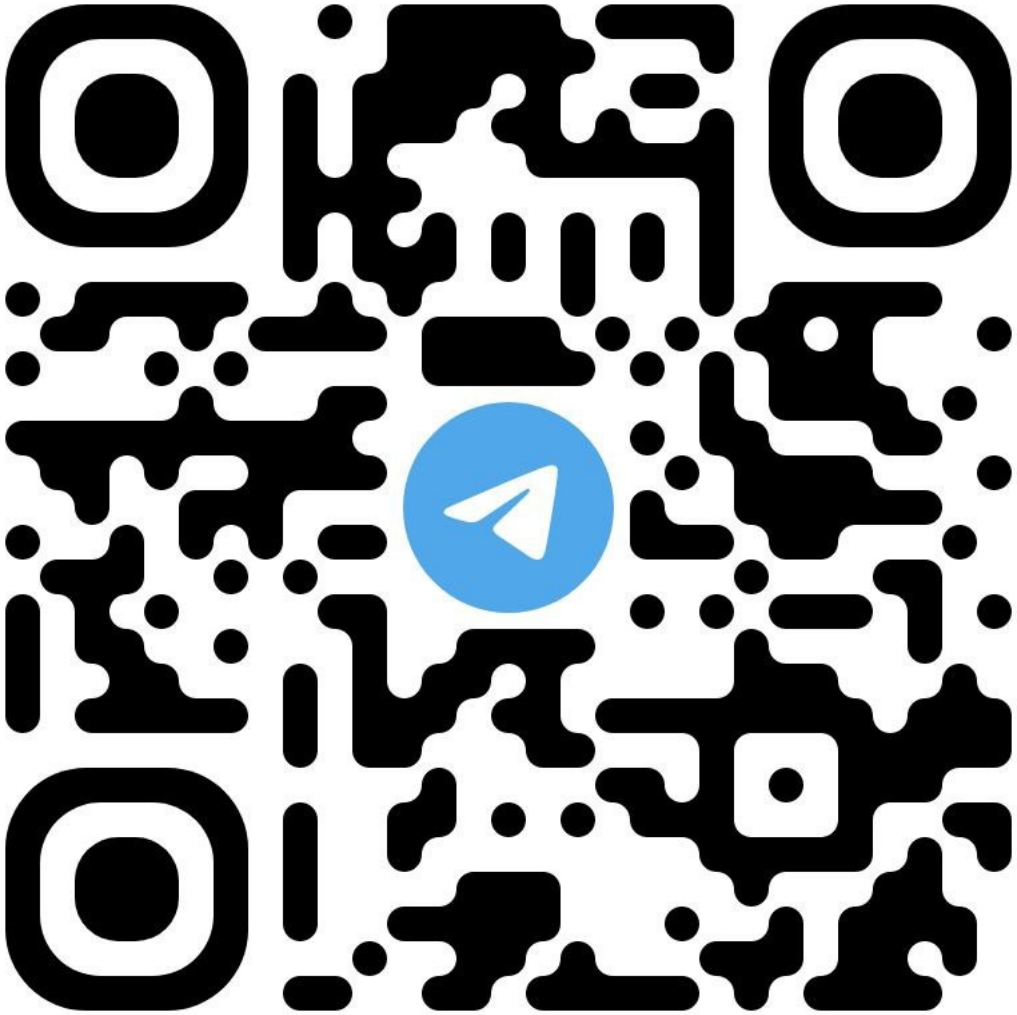
# Deploying ML models



Борисенко Глеб

MLOps 2025

# Немного о себе

- Борисенко Глеб
- Закончил МИФИ в 2019 году, «Информатика и выч. техника», и ФТиАД в 2021 году
- Работаю в Navio в роли ~~MLOps~~ AI-Engineer
- Контакты:
  - E-mail: gleb.brsnko@gmail.com
  - Telegram: @illuser

# О чем курс, критерии оценивания

- Код, его качество, чистота
- CI/CD/CD – Continuous Integration / Continuous Delivery / Continuous Deployment
- Микросервисы, тесты, деплой, мониторинг, пайплайны, данные

- Формула оценки:

$$0.33*дз1 + 0.33*дз2 + 0.33*дз3$$

# Что будет в итоге

- После курса вы будете иметь представление о технологиях, которые помогут автоматизировать ваши тех. процессы
- Куча незнакомых слов перестанет быть таковыми
- Сможете контролировать или хотя бы понимать, как модельки и данные двигаются у вас в работе
- *Добавите в резюме новые слова :3*

# Домашние задания

1. Обернуть модельку в микросервис (реализовать API)
2. Реализовать работу с БД, завернуть в докер, поднять в миникубе (kubernetes с одной нодой)
3. Написать автотесты, прикрутить логирование и мониторинг, настроить CI/CD

*P.S.: Каждое следующее задание, как вы видите, связано с предыдущим, но если вы не захотите какое-то из них делать, будет предложена «болванка» или взять проект друга.*

# Структура курса по темам (Part 1)

1. Что такое MLOps, особенности процесса, роли в команде
2. Работа с кодом: зависимости, культура, как писать хороший код, паттерны, линтеры, форматеры, треды и процессы, IPC
3. Немного о сетях, REST API, gRPC
4. Сервис в python: Flask, FastAPI, Swagger, Streamlit, Bootstrap
5. Версионирование кода и артефактов: Git, gitflow, nexus
6. Виртуализация: Docker, Kubernetes
7. Тестирование: его виды и pytest

# Структура курса по темам (Part 2)

8. Автоматизация тестирования и запуска моделей: Airflow, Kubeflow, ClearML, Gitlab, Jenkins (2 занятия)
9. Версионирование данных: DVC, валидация
10. Мониторинг: Grafana, Prometheus, VictoriaMetrics
11. Потоки данных: Kafka, Flink, Feature Store
12. Развертывание приложений, Ansible + Terraform

# Что почитать

- В чатик чуть позже скину ссылку на гит, где будут появляться все материалы, а также ссылки на что и где почитать
- В основном это разные доки
- Есть классный сайт: ml-ops.org
- Есть классная книжка с кабанчиком: “Designing Data-Intensive Applications”
- За основу структуры курса, а также некоторые материалы взял отсюда: github.com/data-mining-in-action/DMIA_ProductionML_2021_Spring

# Обзор отрасли, что вообще такое этот ваш “Production”

# Типичные задачи с точки зрения теории

- 3 базовые задачи - классификация, регрессия, кластеризация, генерация
- Встречаются все
- Самый частый случай - принятие решений – скорее классификация
- Будем в дальнейшем в голове держать именно случай классификации, но принципиальной разницы нет

# Типичные задачи с точки зрения теории

- Классические задачи:
  - Рекомендательные системы
  - Рекомендательные системы
  - Поисковые системы
  - Churn prediction
  - LTV
  - Uplift modelling
  - Прогноз временных рядов
  - Антифрод (боты)
  - Антифрод (финансовый фрод)
  - Антифрод (запрещенный контент)

# Типичные задачи с точки зрения теории

- Текст
  - Анализ тональности (чаще всего для отзывов)
- Картинки
  - Классификация картинок
  - Сегментация
  - Генерация
- Звук
  - Text-to-speech
  - Speech-to-text

# Оно же, с точки зрения бизнеса

- Улучшать существующий автоматический процесс
  - Находить важные и срочные обращения в тех. поддержку с ML взамен поиска по ключевым словам
  - Улучшить ранжирование в поиске
  - Внедрить ML рекомендации взамен эвристик

- Улучшать существующий ручной процесс
  - Подсказывать тех. поддержке варианты ответа на обращение
  - Подсвечивать подозрительные участки на КТ

- Автоматизировать существующий ручной процесс
  - Автоматизировать клиентскую тех. поддержку
  - Self-driving cars
  - Автоматически проверять 90%+ объявлений на фрод перед публикацией

# Research. Соревнования. Индустрия.

# Свойства ML-решения

Есть разные способы решения задачи, решение может иметь разные свойства, например:

- понятное (для заказчика)
- простое в поддержке
- быстрое (с точки зрения ресурсов при обучении)
- быстрое (с точки зрения ресурсов при инференсе)
- качественное (с точки зрения метрики)
- надежное/робастное (с точки зрения используемого
- подхода, фич и вероятности переобучения)
- научная новизна (самые современные модели, трендовость, неизвестные подходы)
- …

Выделим три основных свойства/“оси”: (научная) новизна, качество, простота. Почему они? Потому что ярче остальных отражают основные подходы.

# Новизна / Research

- Критерий - получился ли новый подход / модель.
- Заранее не известно, заведется ли
- Тут важно научиться решать новые задачи / новые подходы к известным задачам.
- В чистом виде - это академический ресерч.
  - иногда такое есть в крупных компаниях или технологических стартапах

# Качество / соревнования

- Критерий - важна только метрика, пусть даже в ущерб
- всему остальному
- Больше фичей, больше разных моделей, больше стэкинга - это все сюда
- Потолок обычно не известен
- В чистом виде - соревнования
- Как правило, индустриальный рисерч - он именно про это.
  - Про перебор разных комбинаций в рамках задачи.
  - Ближе к соревнованиям, чем к академическом рисерчу

# Простота / индустрия

- Критерий - модель простая в использовании
- Часто это компромисный вариант между качеством и удобством в понимании/дебаге/поддержке
- Наиболее предпочтительный вариант для прода, особенно если моделей много
- Задача MLE - уменьшать техническую сложность и повышать надежность, чтобы как можно больше моделей попадали в эту категорию.

# Примеры

- Банковский скоринг
  - Требуется интерпретируемость
  - Робастность/надежность важнее качества

- Трейдинг
  - Качество на первом месте
  - Никаких ограничений на ресурсы обучения
  - Ограничение на время применения

# Примеры

- Технологический стартап
  - Качество на первом месте
  - Задача может быть плохо изучена
  - Деплоить/обслуживать толком нечего
  - Нет ограничений по ресурсам

- E-Commerce
  - Моделей много, все надо обслуживать
  - Есть продакшн, его ломать нельзя
  - Мир меняется, продукт развивается, модели быстро устаревают
  - Лучше брать модели на 2% хуже, но в 10 раз быстрее/легче в обслуживании

# Стадии проекта как бизнес сущности

- Pre-Sale
- Pilot
- Service Dev
- Service Stable

# Pre-Sale

- Этап оценки технических требований и имеющихся данных
- Попытка найти нишу и возможность сделать продукт
- Часто “теоретический”

# Pilot

- Технический прототип
- Должен работать, но не идеально
- Основная цель - доказать жизнеспособность идеи
- Как правило, нет жестких требований к скорости, качеству или надежности
- На этом этапе важна скорость разработки

# Service Dev

- На этом этапе наращивается стабильность и качество сервиса
- Оптимизируется качество моделей, перфоманс
- Появляются требования к скорости, качеству, надежности

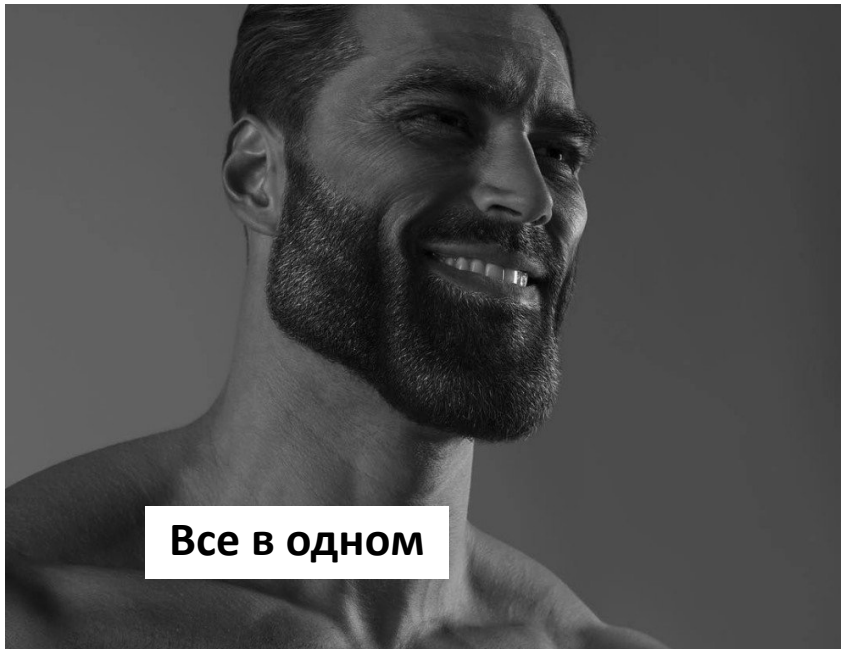
# Service Stable

- Качество моделей уже почти не растет
- Появляются требования к стабильности
- Логи, мониторинги, тесты

# Роли в команде

Рассматриваем только технические. Некоторые роли могут отсутствовать, часто один человек совмещает несколько ролей (и не факт, что это хорошо)

- Data Engineer
- Data Scientist
- ML Engineer
- ML Ops



# Data Engineer

- Делает так, чтобы данные стекались в нужное место в нужном формате
- Следит за наполнением/схемами/миграцией данных
- Следит за технической стороной работы хранилища, помогает профилировать тяжелые запросы.
- Как правило, пошарен между многими проектами
- Функция саппорта хранилища

# Data Scientist

- Смотрит в данные
- Формулирует задачи
- Собирает датасеты
- Обучает модельки
- Центр бизнес-логики проекта

# ML Dev / ML Engineer

- Решает технические проблемы интеграции / перфоманса.
- Превращает модельки в сервис / интегрирует с продуктом.
- Пишет тесты, ищет технические баги.
- Доводит модели до прода.

# ML Ops

- Помогает выкатывать модельки
- Помогает строить пайплйны
- Настраивает логи/мониторинги/алерты
- Автоматизирует процессы
- Дает инструменты. Уменьшает сложность. Обслуживает продукт.

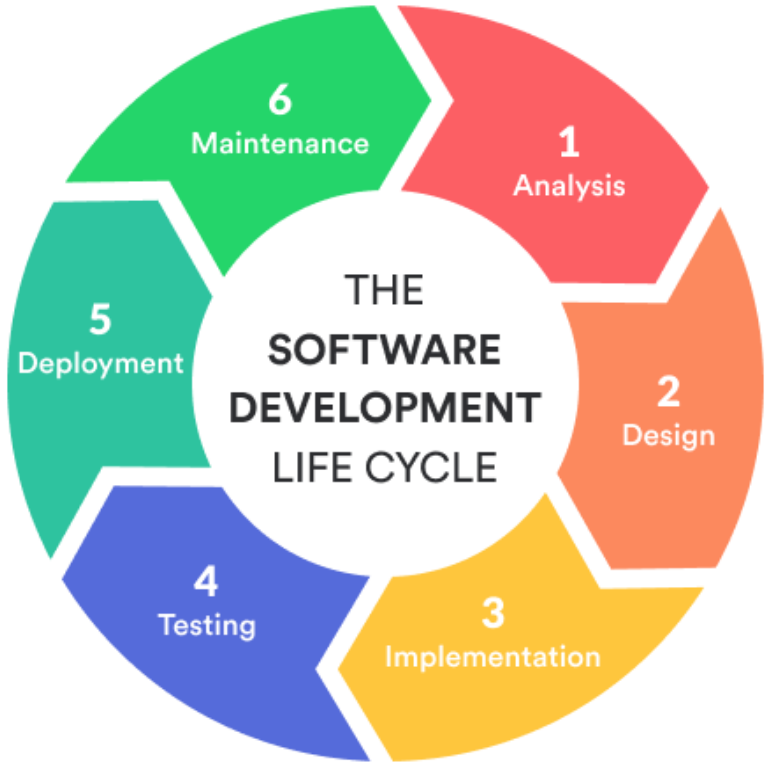
# Примеры

- В больших проектах как правило есть отдельно Data Engineer. Роль DS и ML Dev совмещена - то есть человек занимается постановкой задач, их решением и интеграцией решения в продукт. Роль MLOps берут на себя существующие в компании DevOps.

- В технологических стартапах роль Data Engineer оказывается невостребованной, как и ML Ops. Основной движущей силой является DS и ML Dev.

# Ближе к MLOps

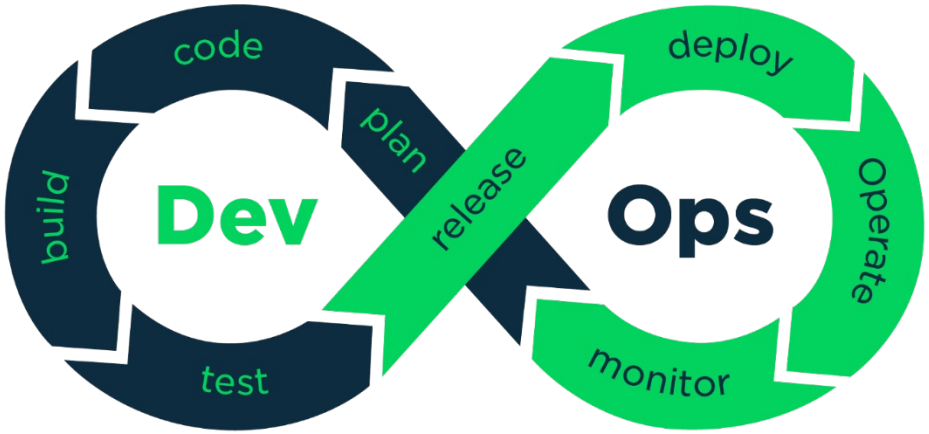
# SDLC


6
Maintenance
1
Analysis
5
Deployment
THE
SOFTWARE
DEVELOPMENT
LIFE CYCLE
2
Design
4
Testing
3
Implementation

Source

# DevOps

DevOps - набор практик, объединяющий разработку ПО (Dev) и управление IT-инфраструктурой и сервисами (Ops). Цель DevOps - сократить цикл разработки и обеспечить непрырывную поставку обновлений (Continious Delivery) с высоким качеством.




Source

# Ключевые сферы DevOps

- Coding – code development and review, source code management tools, code merging.
- Building – **continuous integration** tools, build status.
- Testing – **continuous testing** tools that provide quick and timely feedback on business risks.
- Packaging – artifact repository, application pre-deployment staging.
- Releasing – change management, release approvals, release automation, **continious deployment** tools.
- Configuring – infrastructure configuration and management, infrastructure as code tools.
- Monitoring – applications performance monitoring, end-user experience.

# Отличие ML систем от прочих программных

ML системы отличаются от прочих програмных систем в следующем:

- Квалификация команды: DS и MLRs могут не иметь опыта построения production-сервисов
- Разработка: ML-модели требуют постоянных экспериментов.
- Сложность состоит в отслеживании результатов и их воспроизводимости, код должно быть легко изменять и переиспользовать.
- Тестирование: помимо обычного тестирования, нам требуется также проверять качество данных, качество обученных моделей и адекватность их работы.
- Деплоймент: требует задеплоить не просто ML модель, а целый ML pipeline, который уже будет переобучать и деплоить модель
- Production: в отличие от обычного ПО, ML модели устаревают, поскольку данные непрерывно меняются

# Отличие MLOps от DevOps

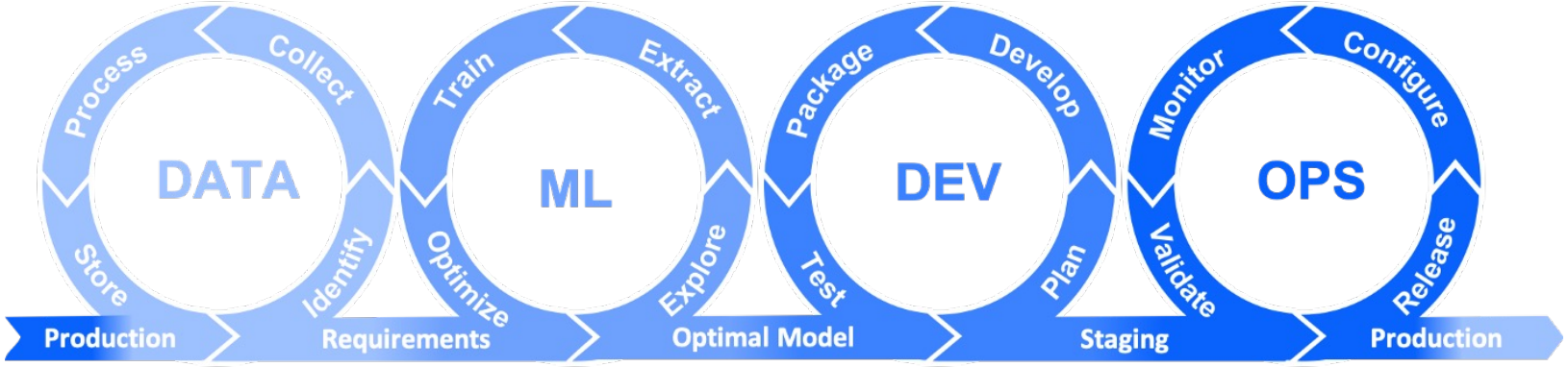
ML and other software systems are similar in continuous integration of source control, unit testing, integration testing, and continuous delivery of the software module or the package. However, in ML, there are a few notable differences:

- CI is no longer only about testing and validating code and components, but also testing and validating data, data schemas, and models.
- CD is no longer about a single software package or a service, but a system (an ML training pipeline) that should automatically deploy another service (model prediction service).
- CT is a new property, unique to ML systems, that’s concerned with automatically retraining and serving the models.

Source

# MLOps Cycle

MLOps is the extension of the DevOps methodology to include Machine Learning and Data Science assets as first-class citizens within the DevOps ecology




Source

# Что должен делать MLOps?

- Стремиться объединить релизный цикл ML и релизный цикл ПО
- Автоматизировать тестирование ML артефактов
- Применять Agile принципы к ML проектам
- Интегрировать ML артефакты в CI/CD системы
- Снижать технический долг, возникающий при использовании ML

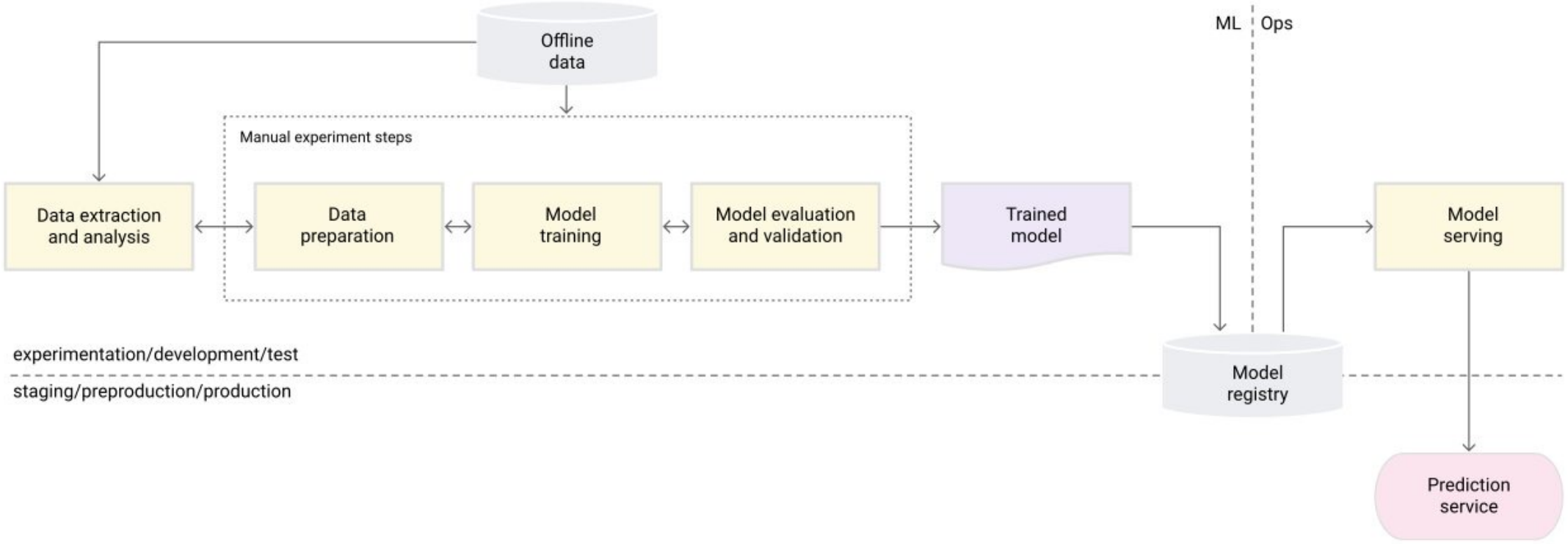
# Немного про техдолг

Технический долг — накопленные в программном коде или архитектуре проблемы, связанные с пренебрежением к качеству при разработке программного обеспечения и вызывающие дополнительные затраты труда в будущем.

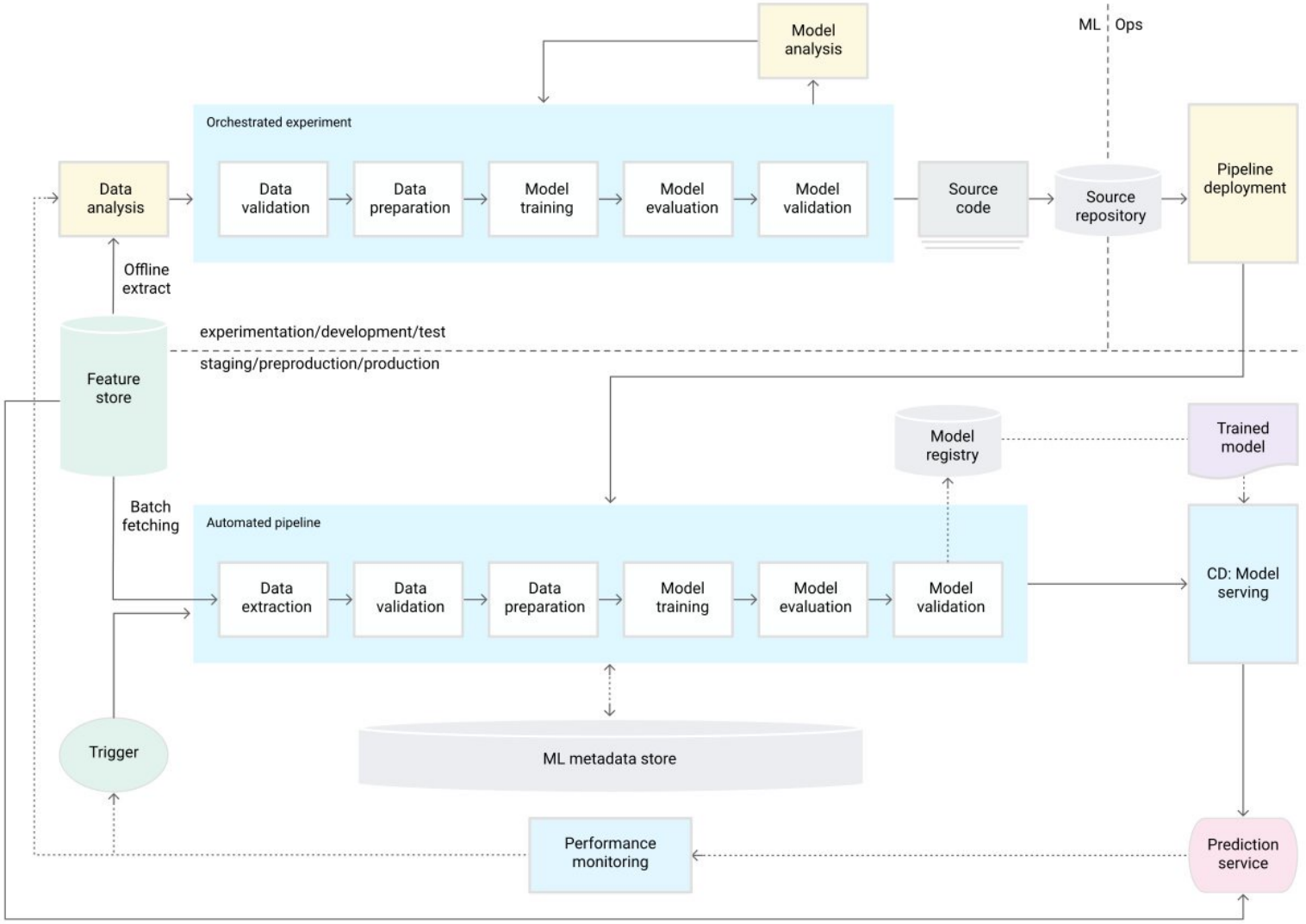
ML-системы имеют дополнительные “возможности” для создания технического долга. ML Engineer сталкивается с этим в первую очередь.

Например — unstable data dependencies, feedback loops, glue code, pipeline jungles, dead experimental codepaths, fixed thresholds in dynamic systems, entanglement (CACE principle), etc.

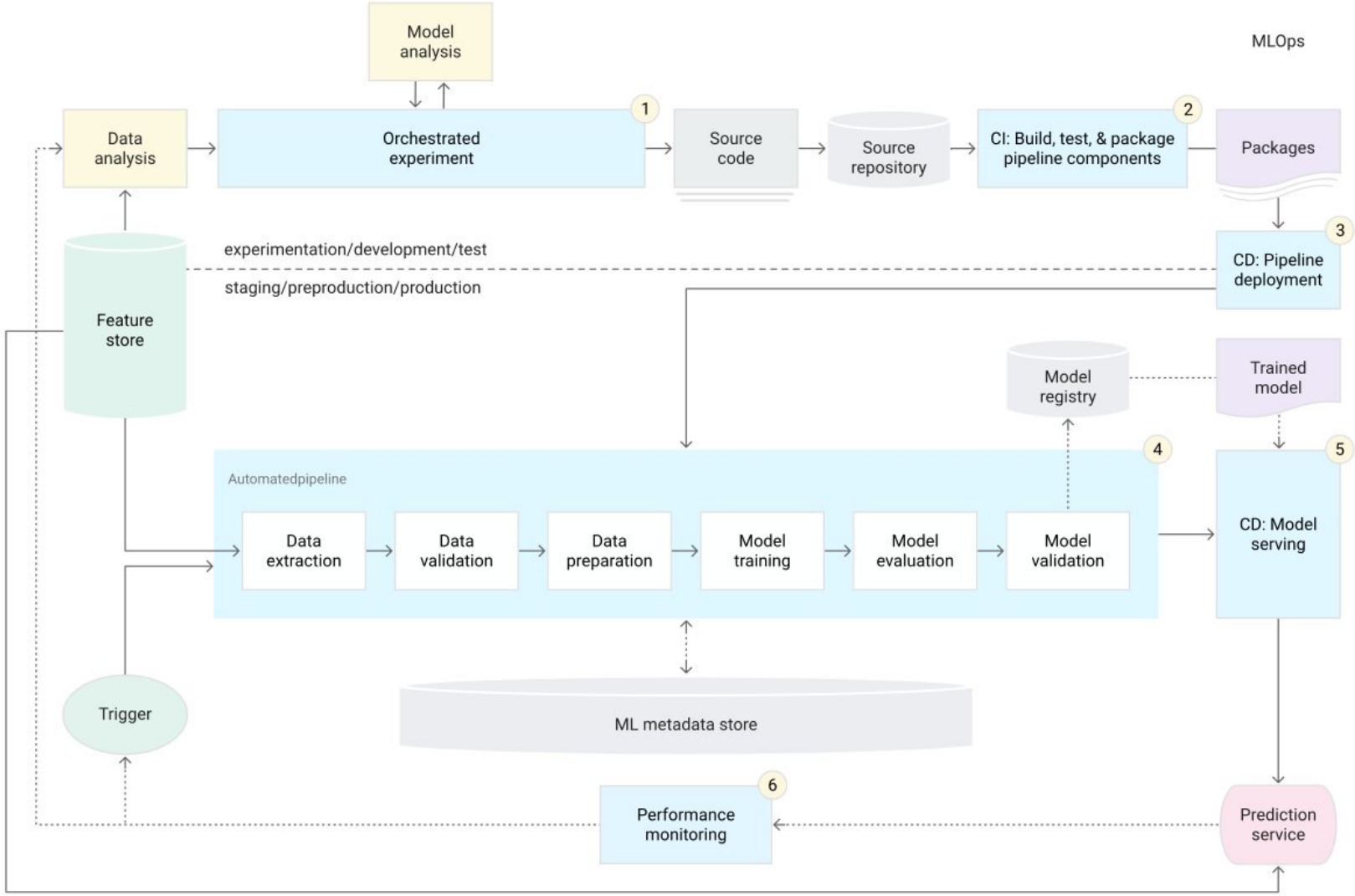
# MLOps level 0: Manual process



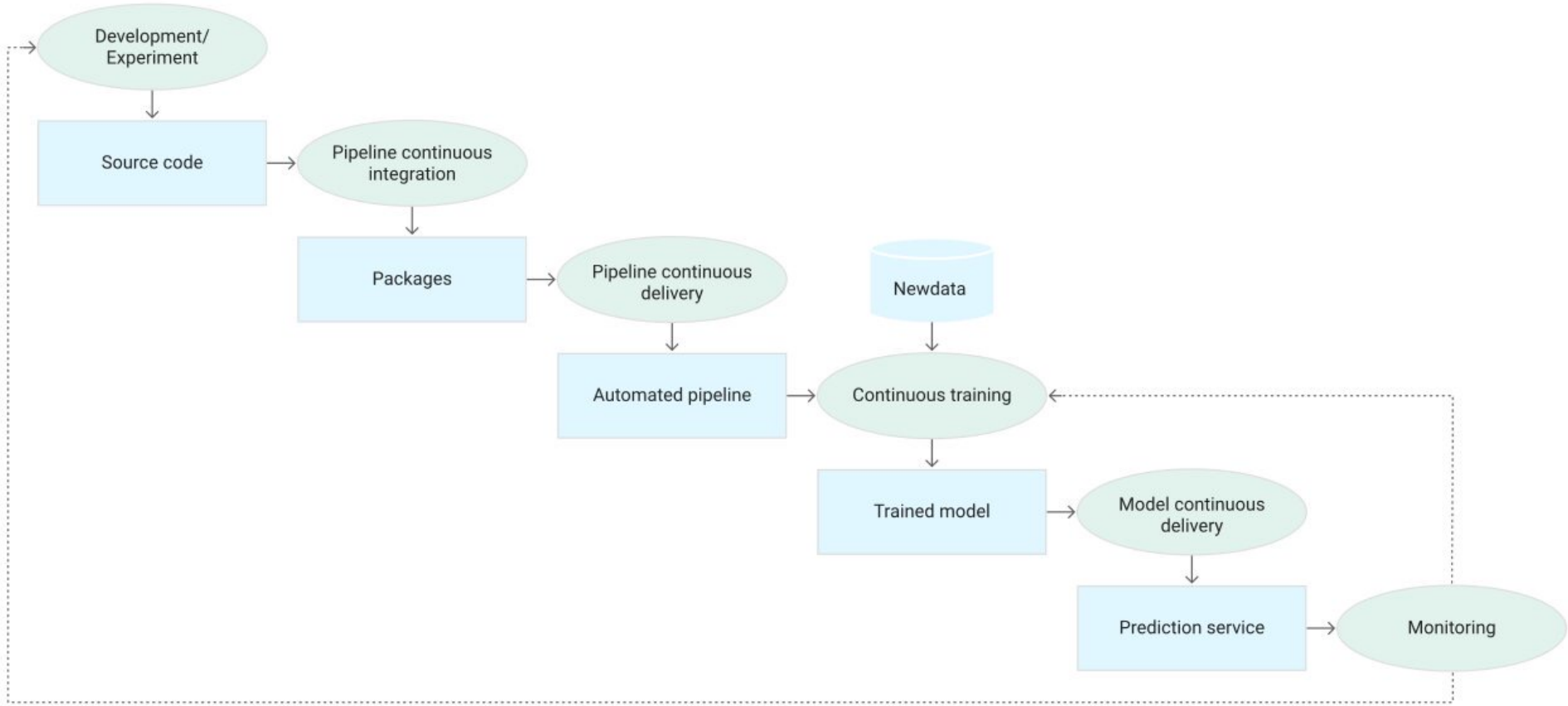
# MLOps Level 1: автоматизация ML пайплайна



# MLOps Level 2: автоматизация CI/CD пайплайна



# MLOps - итоговый пайплайн



# Саммари

1. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.
2. Одними из наиболее важных практик DevOps являются Continious Integration и Continious Delivery.
3. MLOps применяет подход DevOps для разработки ML-систем с учетом их особенностей. С необходимостью обновлять модели добавляется практика Continious Training.
4. В нашем курсе мы будем работать с инструментами, позволяющими реализовать подход MLOps в работе над вашими проектами.

# Model Governance

# Что это такое?

- По сути, это контроль модели
- В вашей модели должна быть уверенность
- Как можно доверять тому, что слеплено из каках и палок?

# На что смотрим?

- Данные – мы должны понимать, откуда они идут, какие преобразования проходят; мы должны уметь распутать клубочек
- Код – тут очевидно
- Веса – также, только мы должны хранить все, из чего получается модель; дообучаем – есть исходные веса; также должны уметь распутать клубочек
- Метрики – их мы тоже должны контролировать, у нас должна быть история их динамики

# Golden …

- Golden Datasets – эталонные датасеты, валидационные; есть также история, это ведь не постоянная штука
- Golden Metrics – «паспорт» нашей модели, метрики, полностью описываемые для нас модель; то, по чему мы сможем определять полностью, подходит ли нам модель

*P.S.: метрики не толья для модели могут быть :3*

# Ответственность

- Вы – разработчики модели, важные люди
- Ваша модель может решать судьбы
- Модель – это черный ящик, поэтому мы должны контролировать все по-максимому
- Как вы сами можете быть уверены в модели, сделанной из хз чего?

# Пожелания

- Дз
- Семинары, практика
- Темы
- Что-то ещщщщё