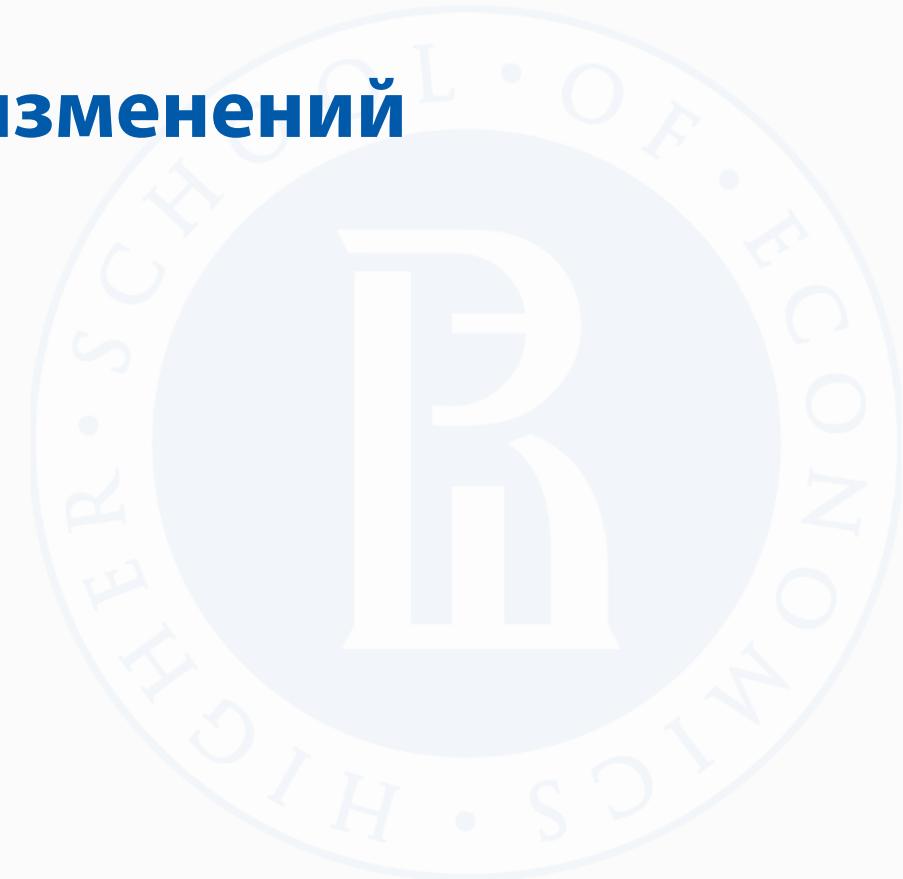


# **Процесс ввода изменений в эксплуатацию**



# План



Требования к процессу введения в эксплуатацию



Воспроизводимость и версионирование производимых артефактов



Приемка изменений: шаги для ввода модели в эксплуатацию с минимальными рисками



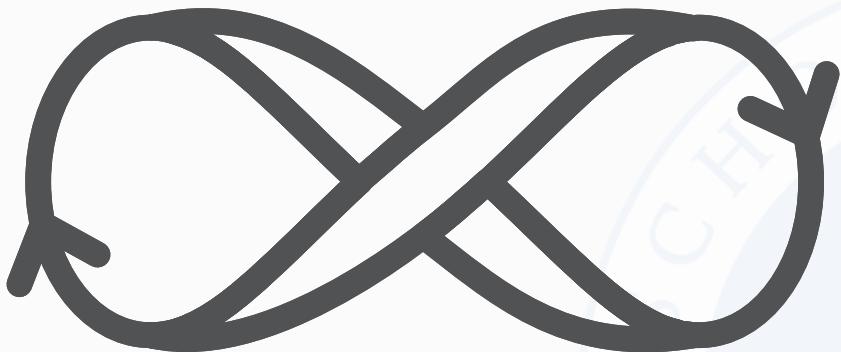
Выкладка и планирование ввода в эксплуатацию



Выкладки с несовместимыми изменениями и миграции данных

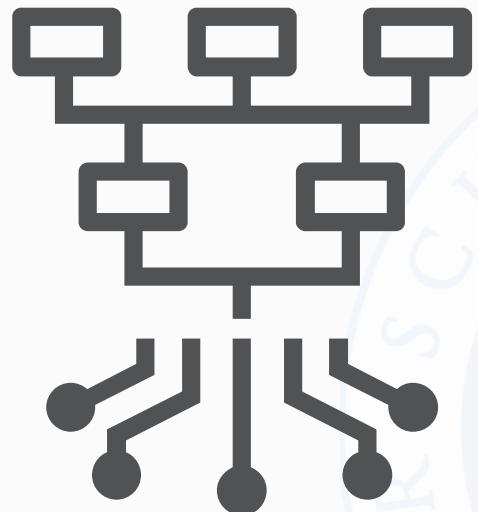
# DevOps и MLOps

DevOps — популярная методология эффективного введения обновлений программ в эксплуатацию

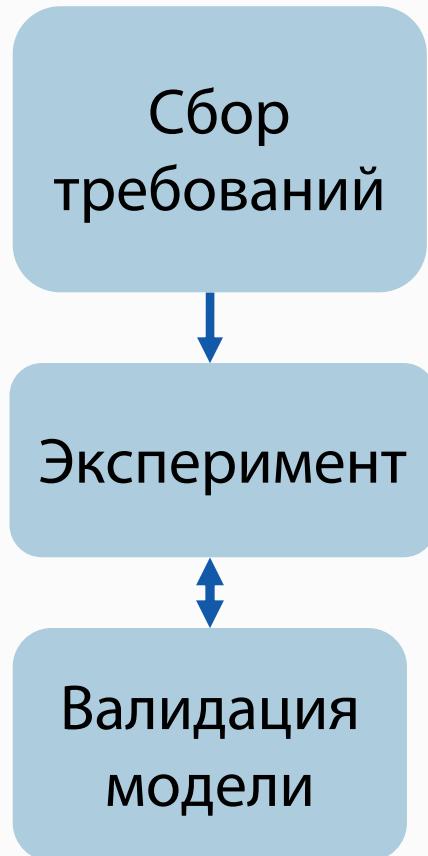


# DevOps и MLOps

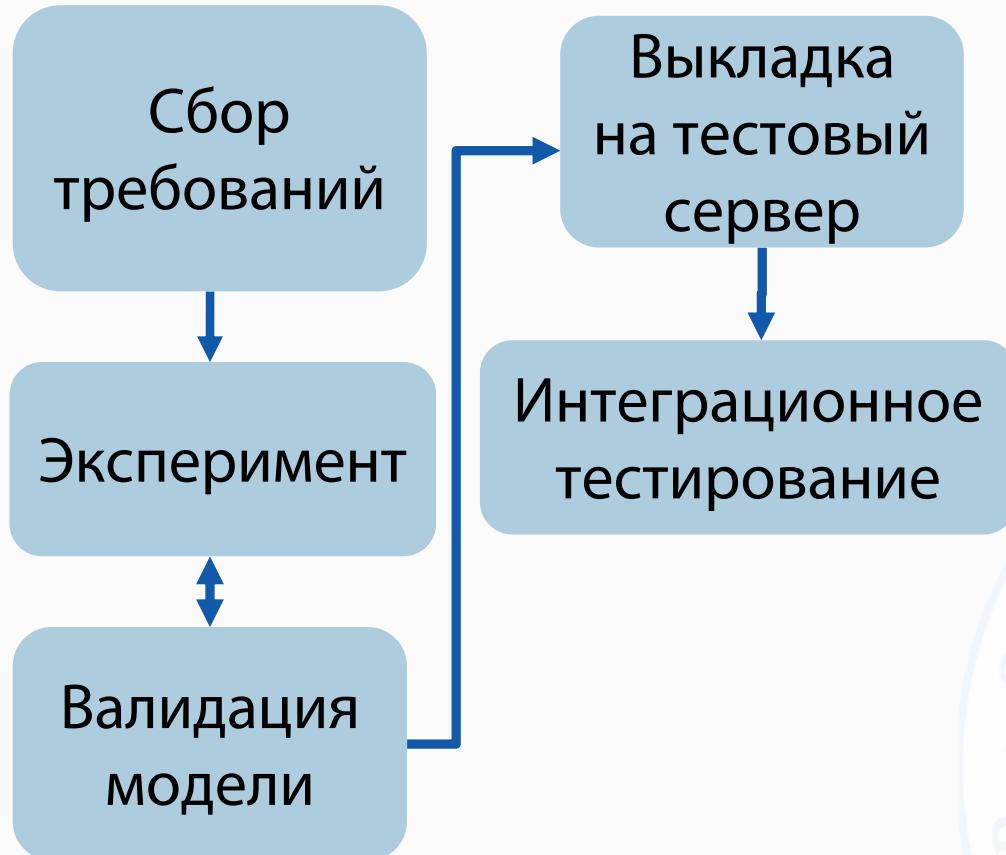
MLOps — методология эффективного введения обновлений ML-моделей в эксплуатацию



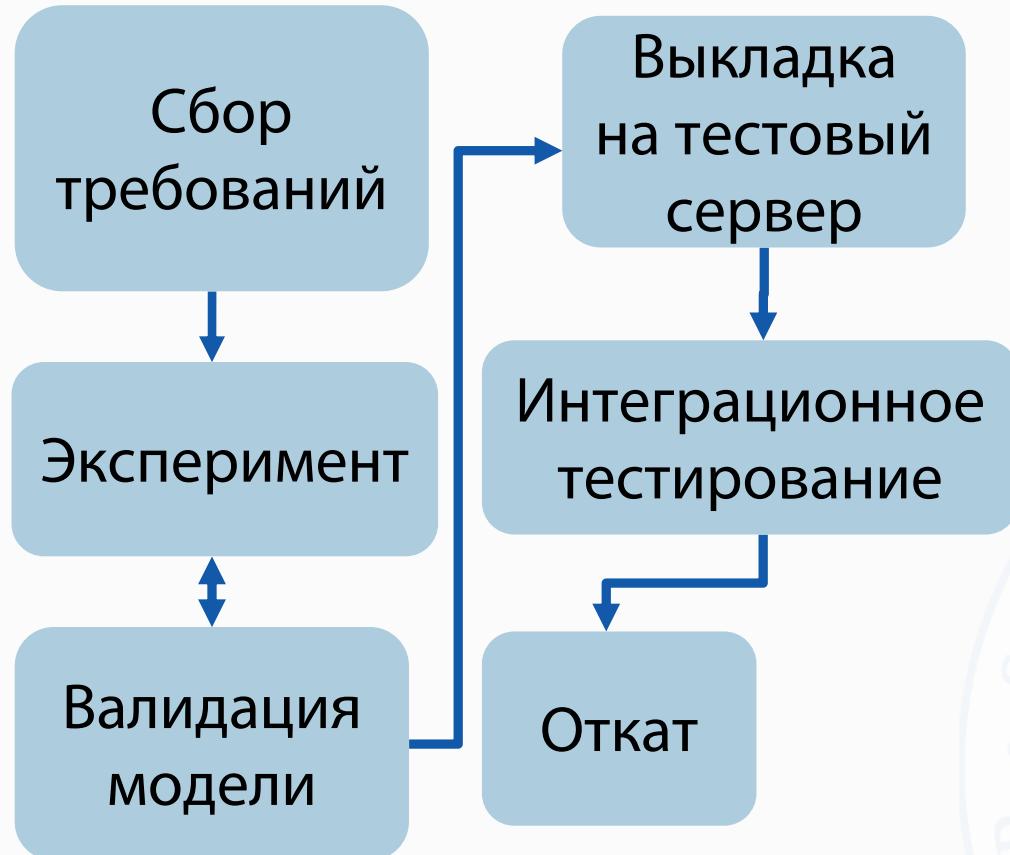
# Полный цикл ввода в эксплуатацию



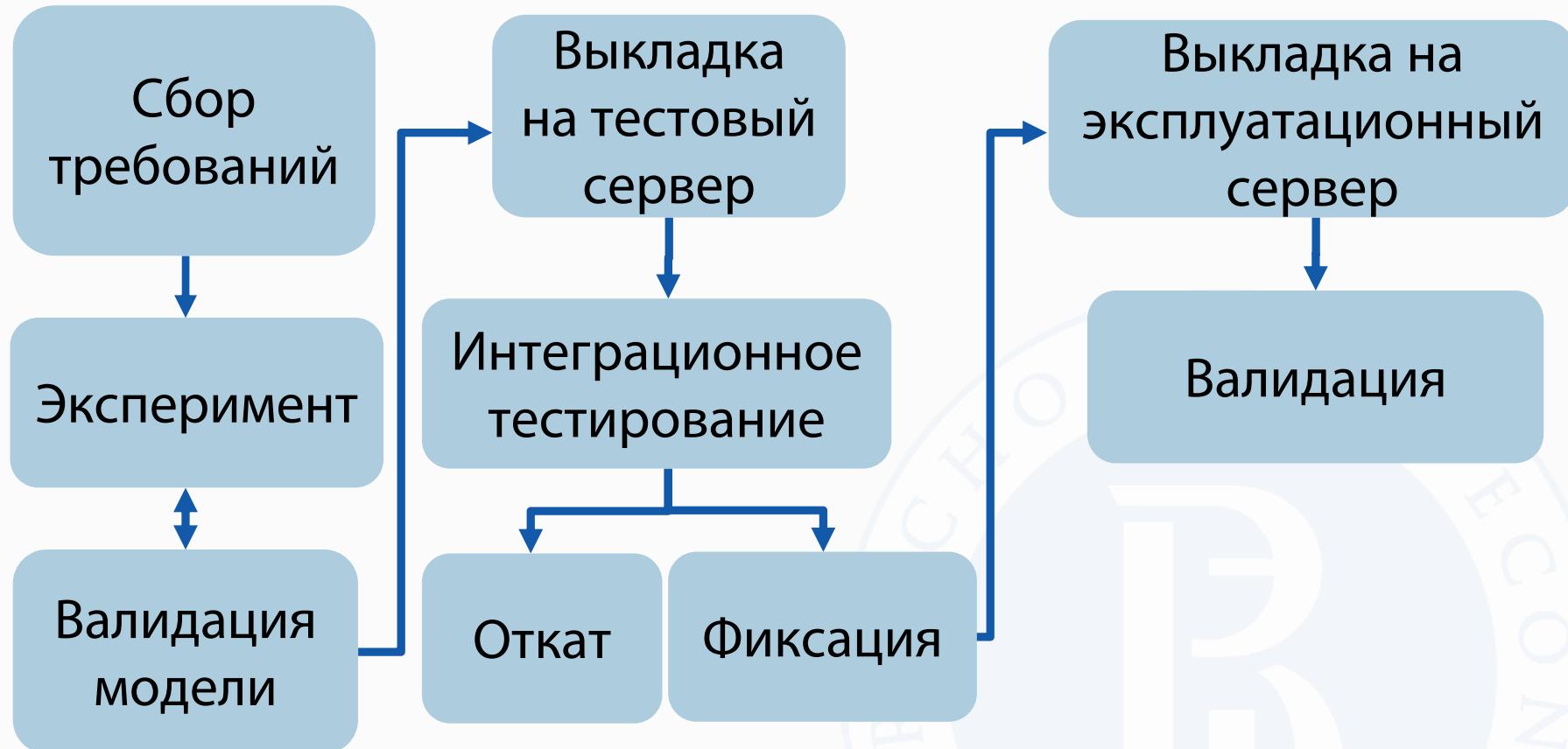
# Полный цикл ввода в эксплуатацию



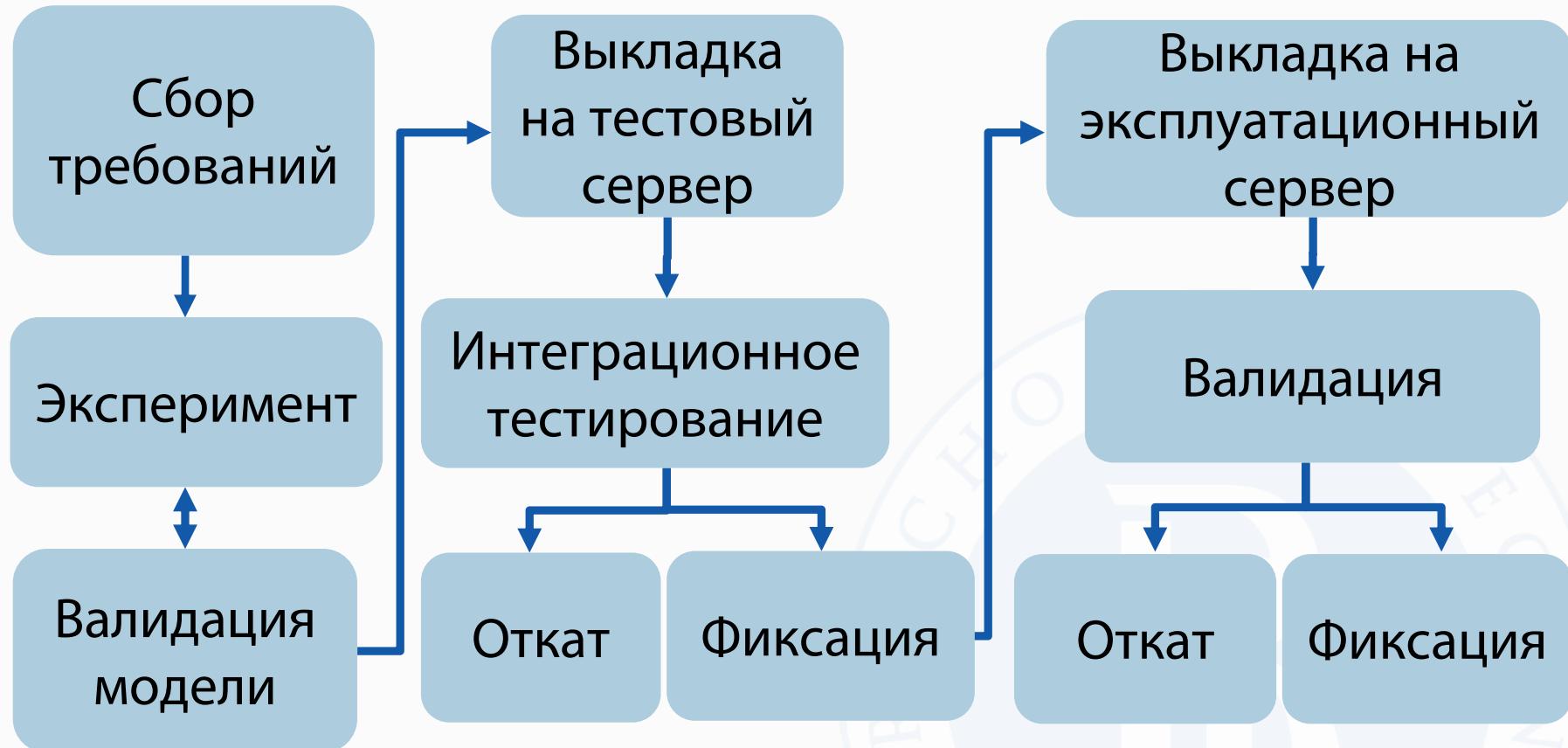
# Полный цикл ввода в эксплуатацию



# Полный цикл ввода в эксплуатацию



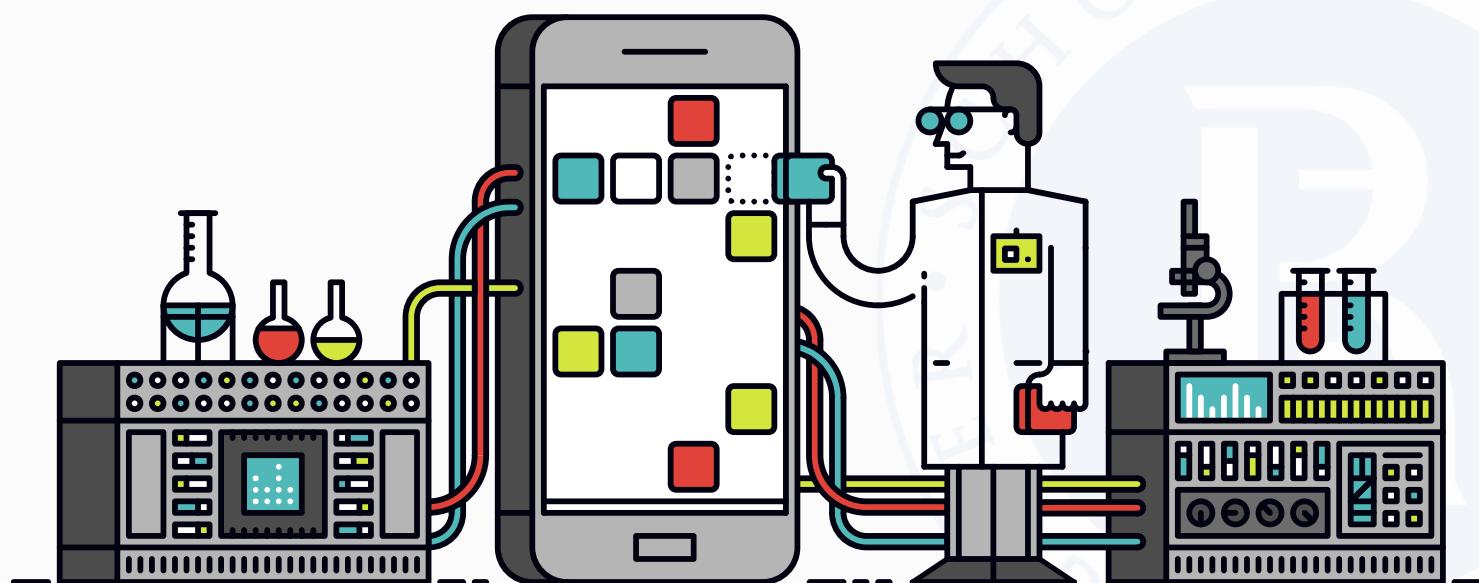
# Полный цикл ввода в эксплуатацию



# DevOps и MLOps

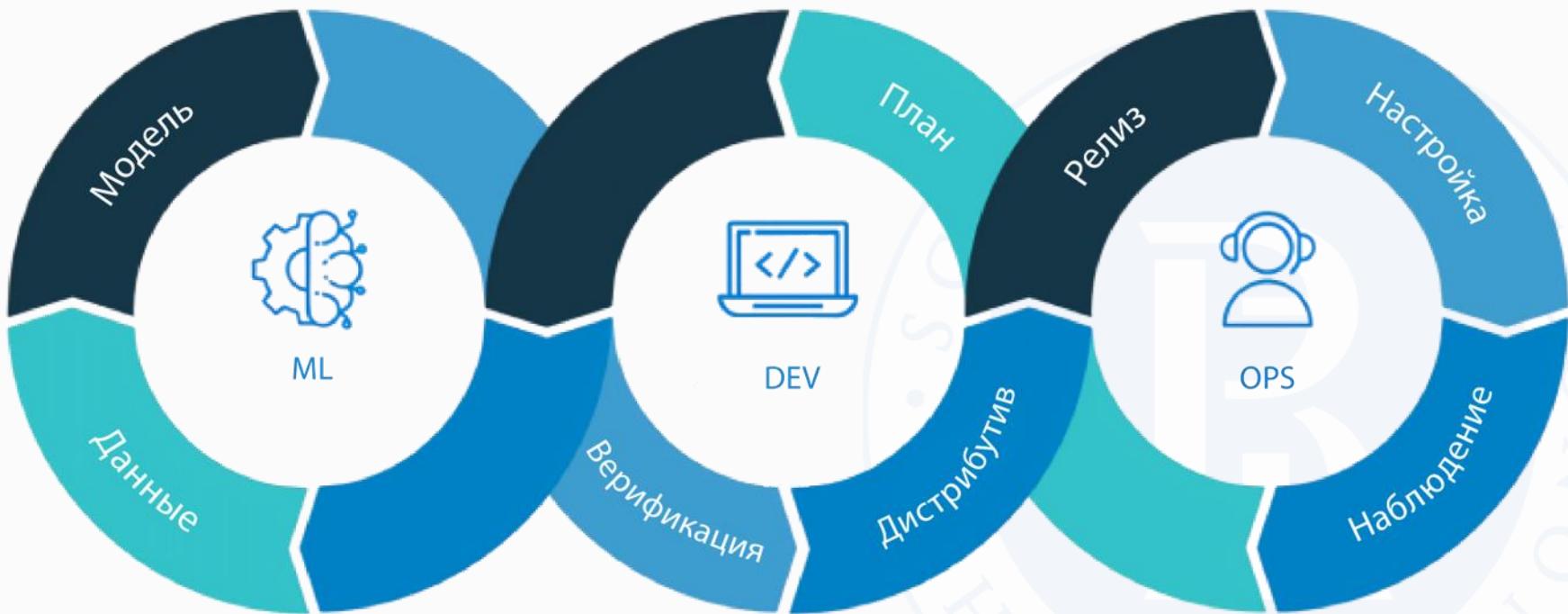
→ Многообразие параметров, что увеличивает шанс запутаться в экспериментах

→ Ключевые особенности:  
Работа с бинарными файлами, которые следует аннотировать, чтобы понимать, чего от них ожидать



# MLOps

- MLOps обеспечивает полный цикл работы с моделями
- Хорошо выстроенный процесс MLOps ⇒ Эффективная разработка и внедрение ML-сервисов



# **Требования к вводу в эксплуатацию**



# План



Архитектура ML-сервиса

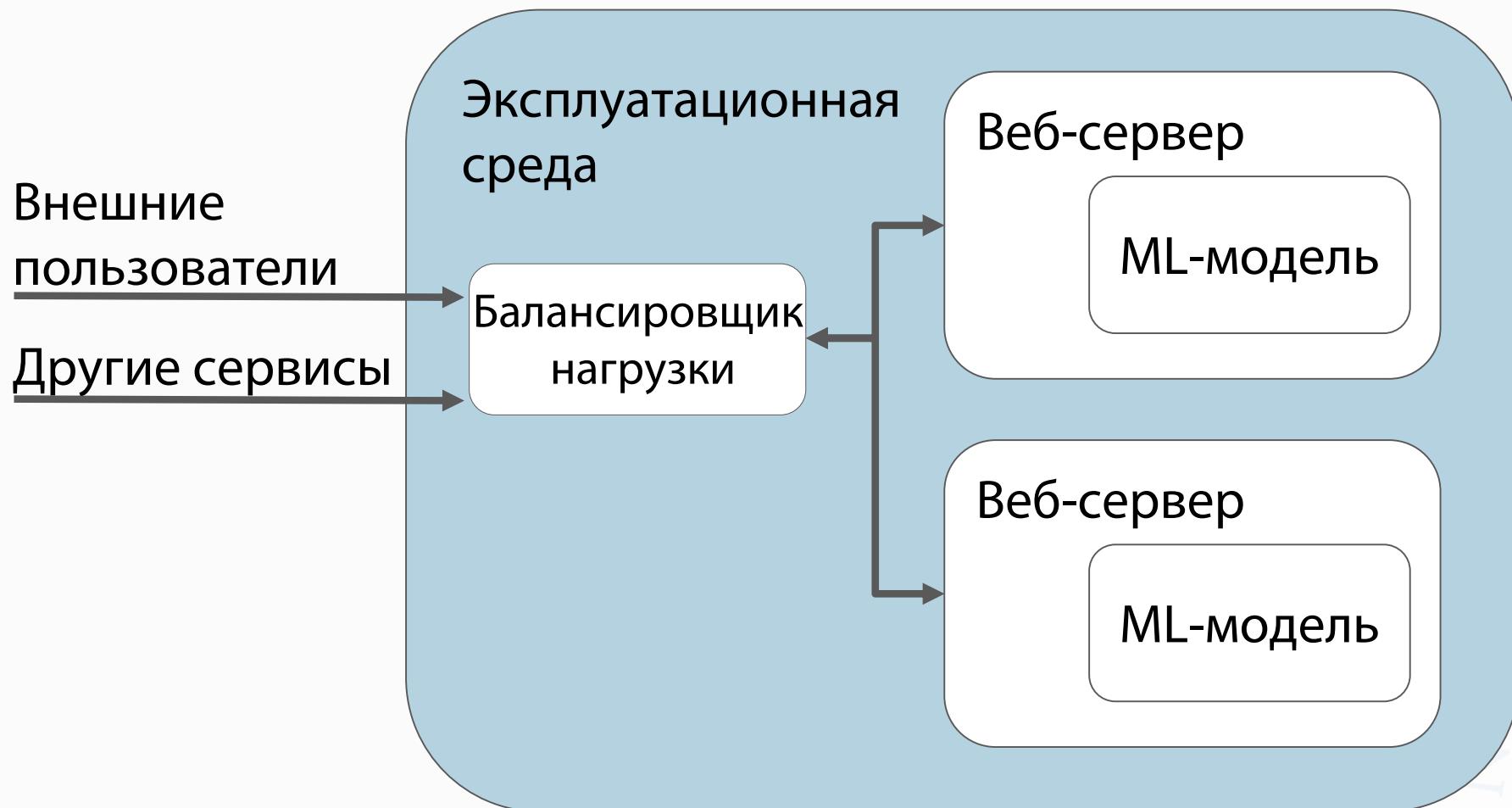


Принципы внедрения изменений

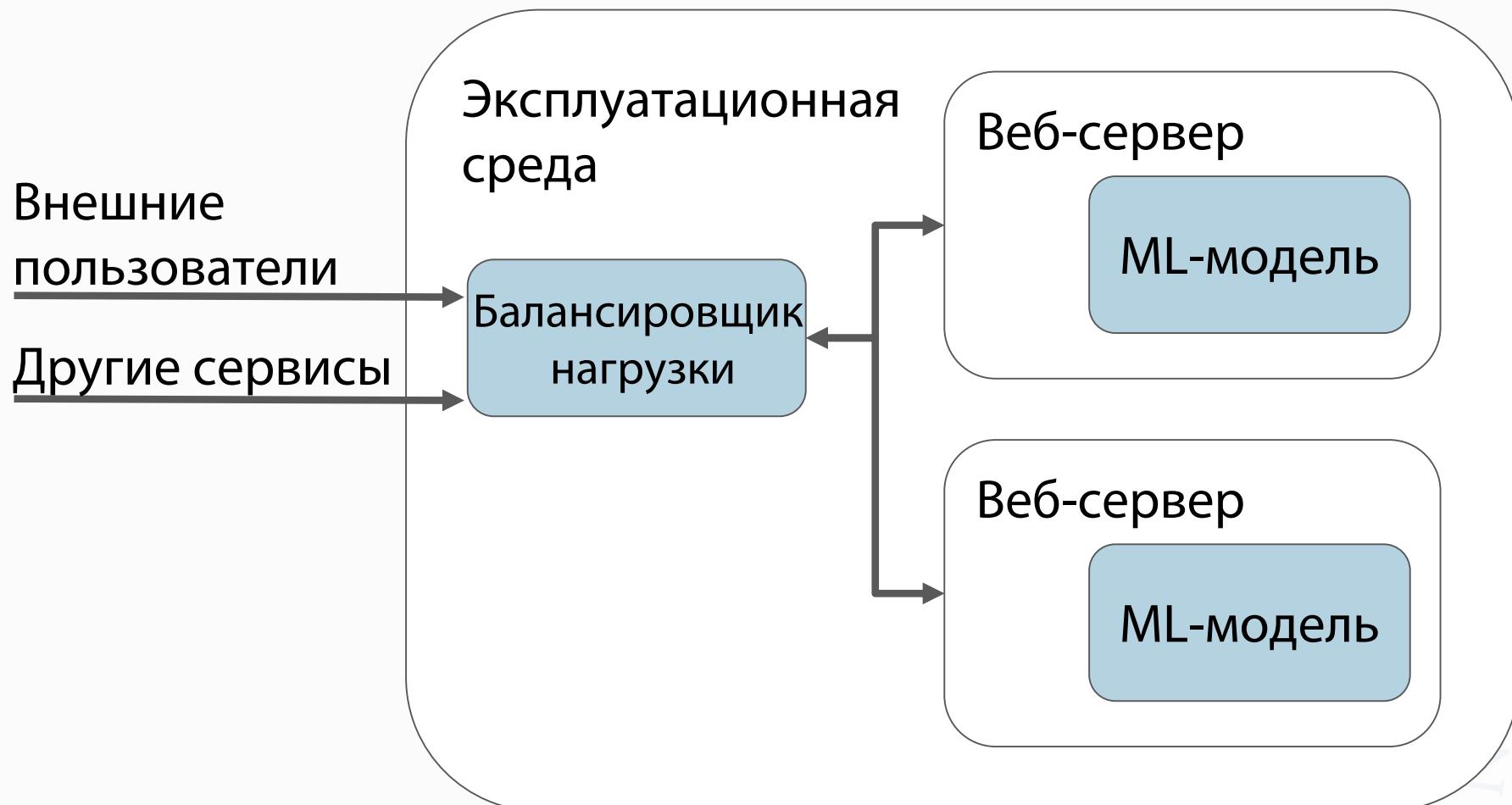


Требования к вводу в эксплуатацию

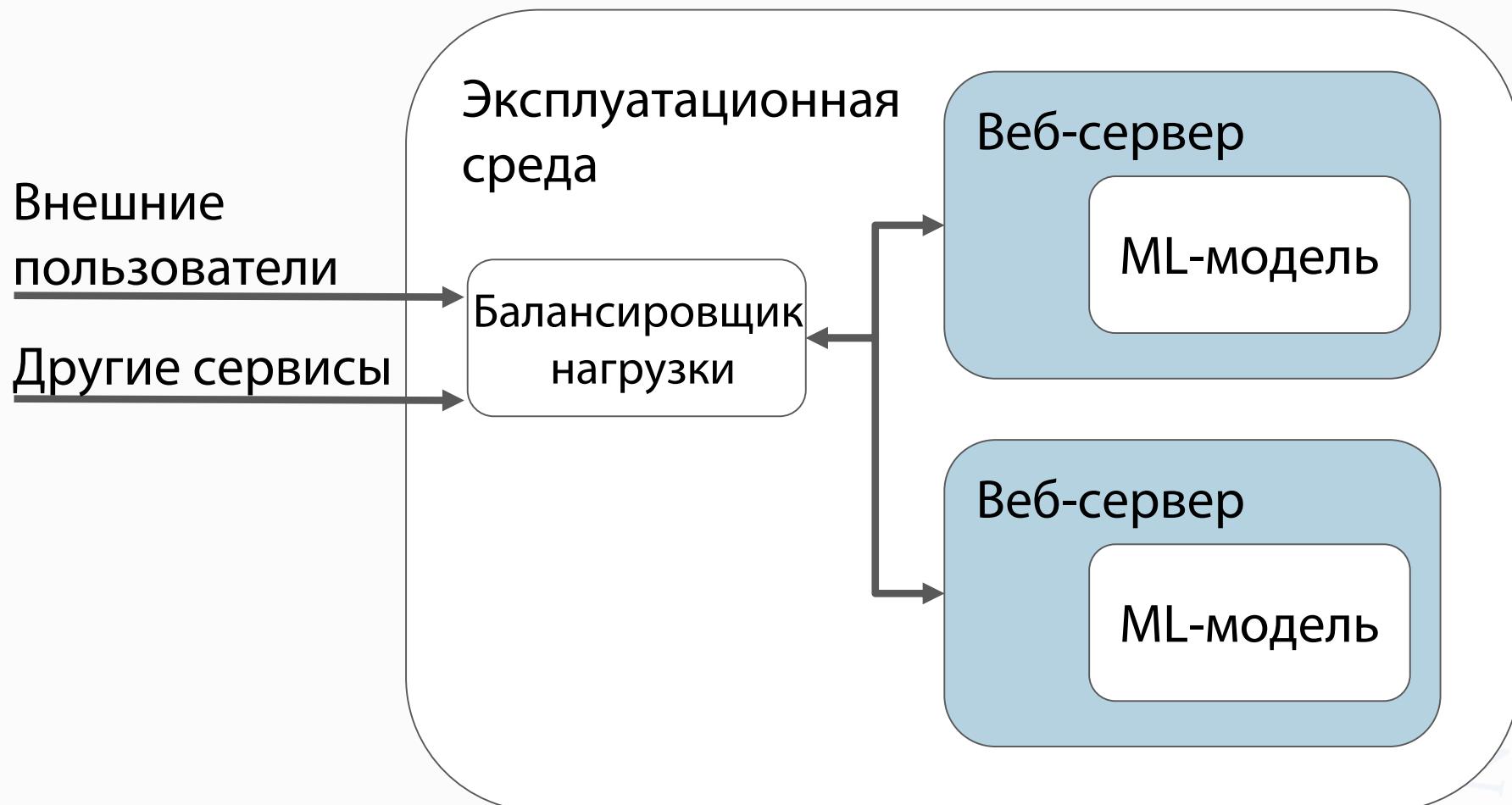
# Архитектура типового-ML сервиса



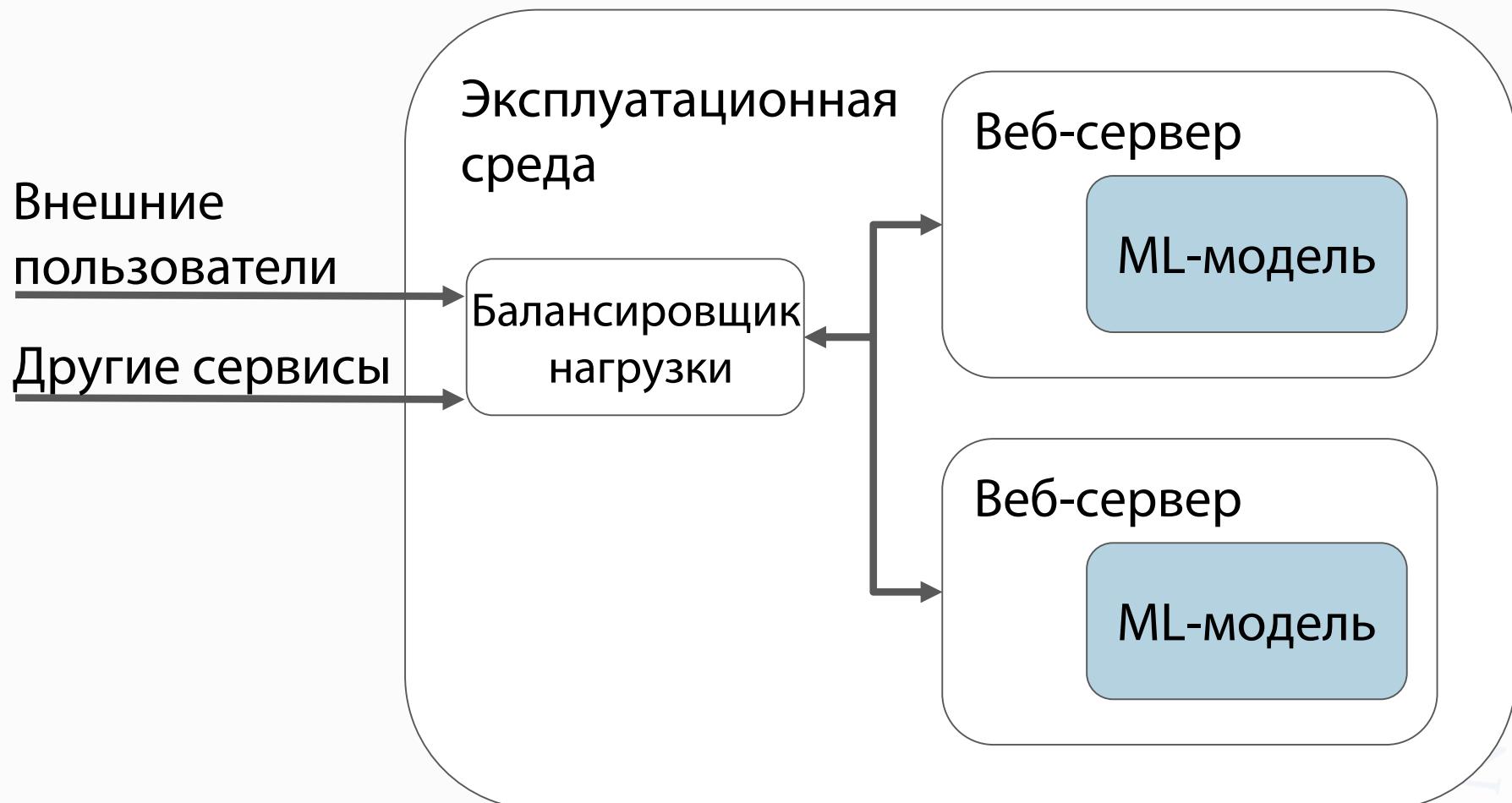
# Архитектура типового ML-сервиса



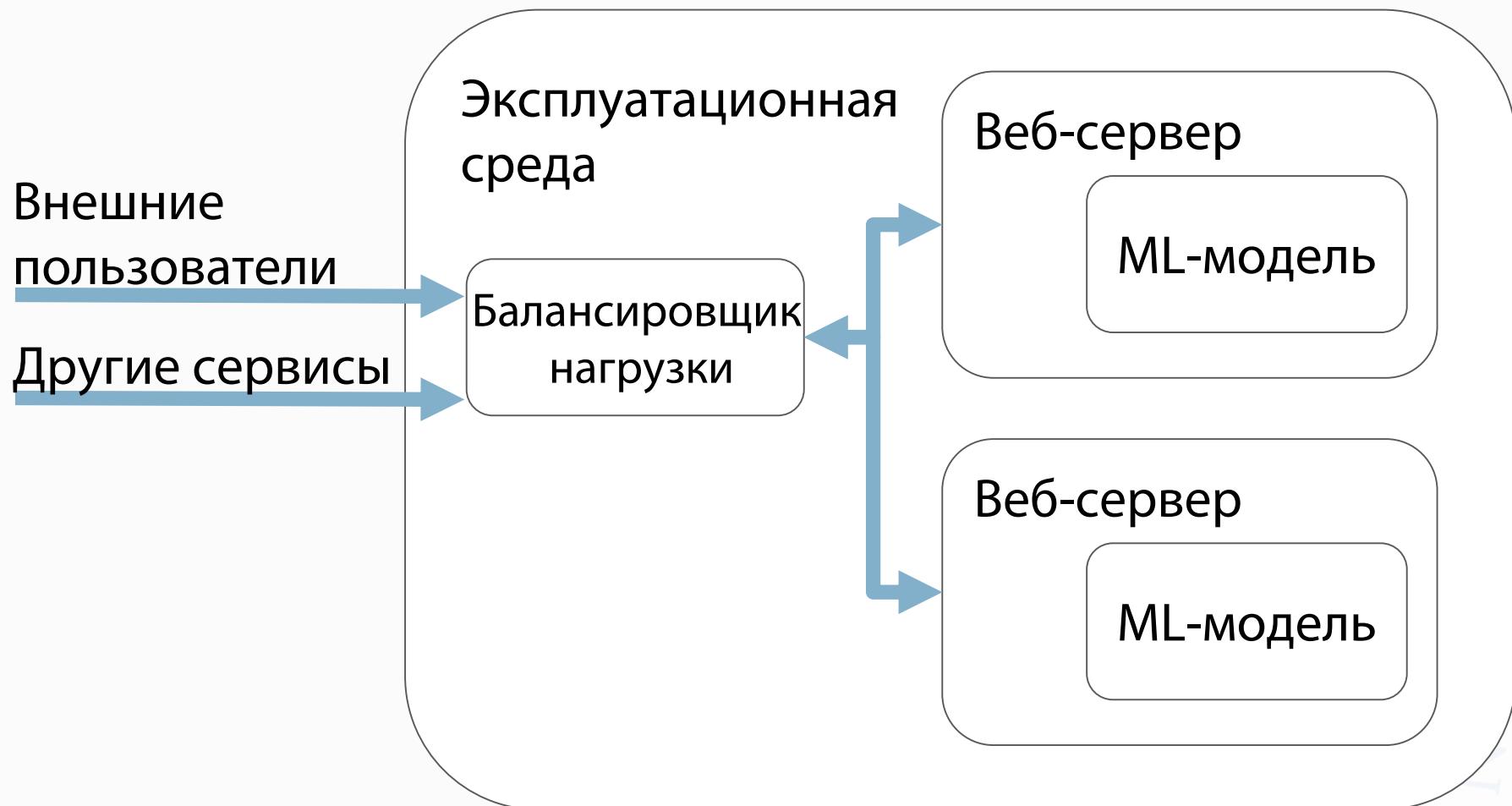
# Архитектура типового ML-сервиса



# Архитектура типового ML-сервиса



# Архитектура типового ML-сервиса



Балансировщик нагрузки опционален,  
если используется один веб-сервер,  
возможна асинхронная схема

# Термины процесса

Артефакт:  
ML-модель

# Термины процесса

## Окружение:

- переменные
- параметры
- зависимости

Артефакт:  
ML-модель

# Термины процесса

## Дистрибутив [1.0.0]

### Окружение:

- переменные
- параметры
- зависимости

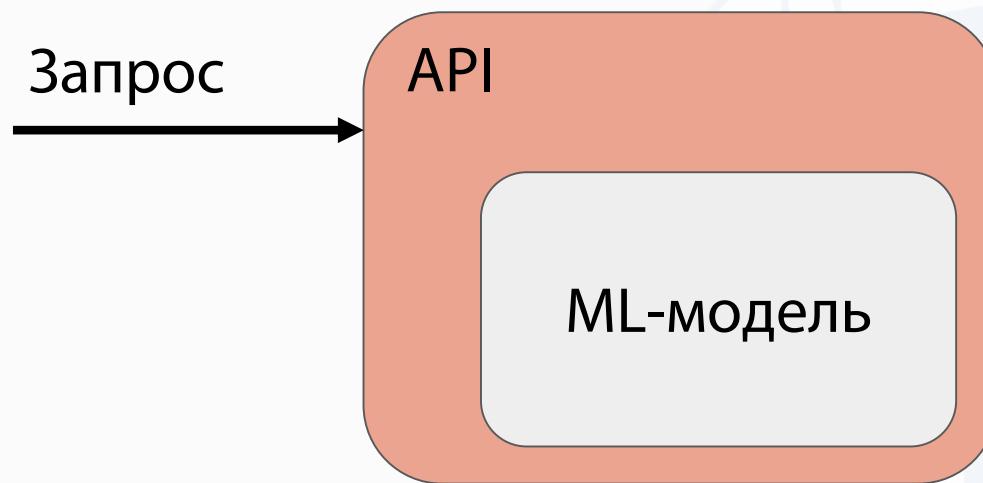
Артефакт:  
ML-модель



Реестр дистрибутивов

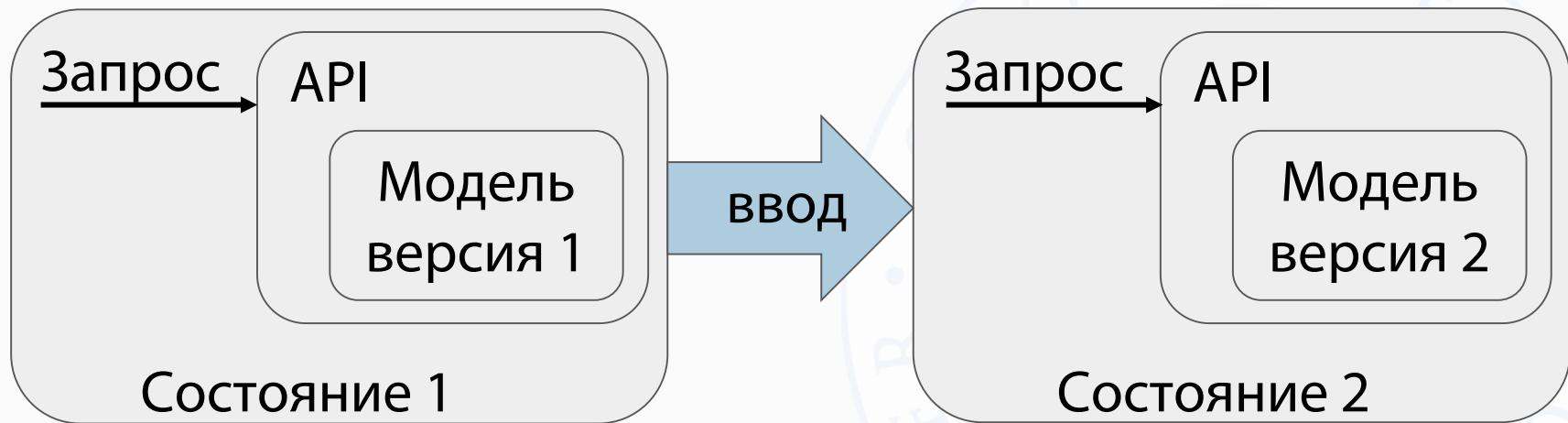
# Ввод модели в эксплуатацию

→ Ввод в эксплуатацию — выкладка дистрибутива  
для использования конечными пользователями через API



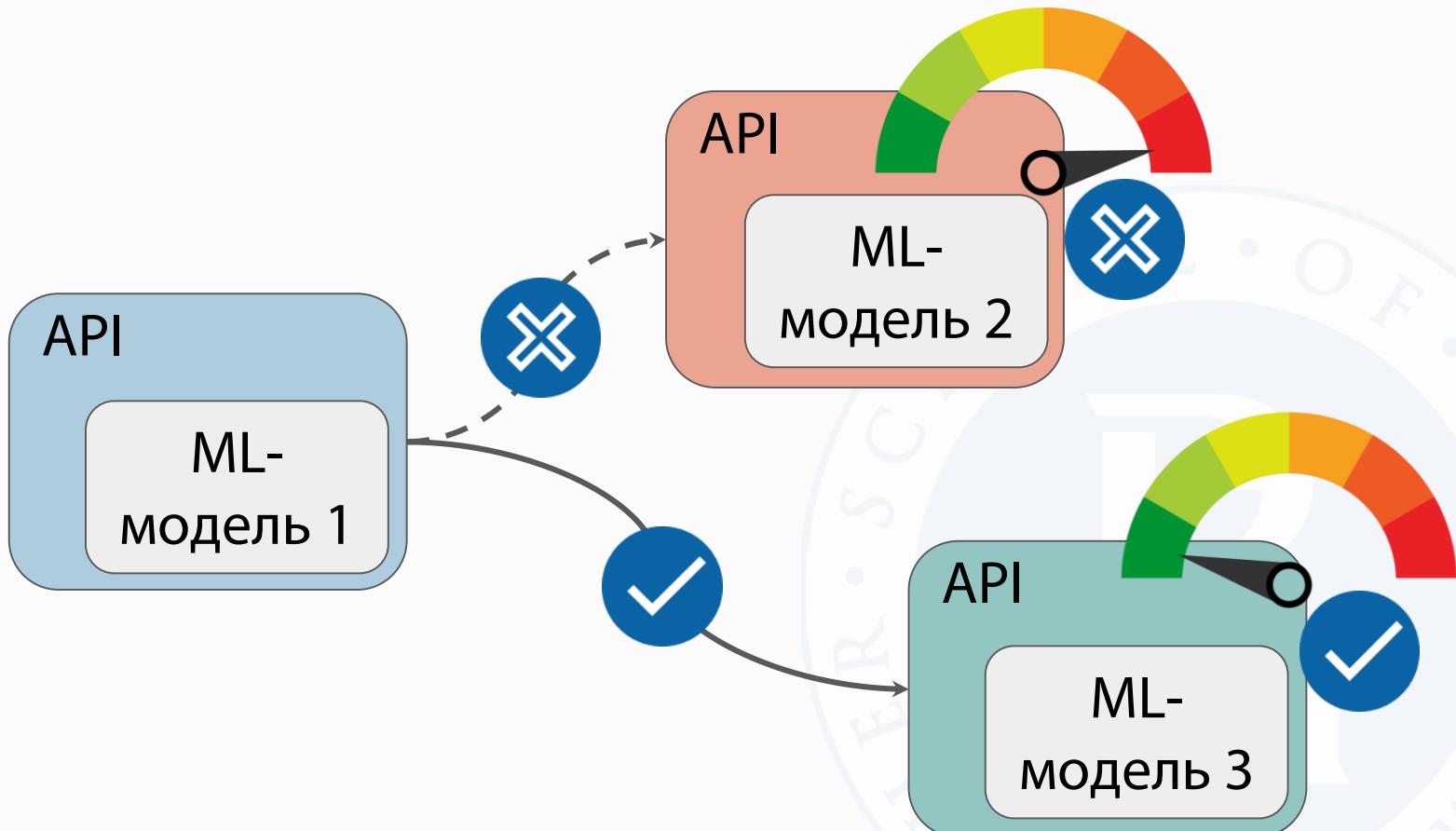
# Ввод модели в эксплуатацию

→ Процесс ввода в эксплуатацию — описание процедуры, в результате которой модель вводится в эксплуатацию



# Ввод модели в эксплуатацию

→ Требования к процессу ввода в эксплуатацию снижают риск выкладки некорректной версии модели



# Примеры ввода некорректных версий

- Введена не та модель, которую планировалось ввести в эксплуатацию
- Введенная модель не работает в окружении эксплуатационного сервера
- Введенная модель работает, но со временем потребление памяти неожиданно растет



# Ключевые вопросы

- Какая модель сейчас работает на эксплуатационном сервере?
- Как оценить, насколько хорошо работает модель на эксплуатационном сервере?
- Каков процесс верификации модели, используемой на эксплуатационном сервере?



# Что сейчас на эксплуатационном сервере?

→ Как описать то, что сейчас работает на эксплуатационном сервере?

# Что сейчас на эксплуатационном сервере?

- Как описать то, что сейчас работает на эксплуатационном сервере?
  - Идентификатор используемой модели

# Что сейчас на эксплуатационном сервере?

- Как описать то, что сейчас работает на эксплуатационном сервере?
  - Идентификатор используемой модели
  
- Есть ли метрики, характеризующие текущее состояние эксплуатационного сервера? Какие значения?

# Что сейчас на эксплуатационном сервере?

- Как описать то, что сейчас работает на эксплуатационном сервере?
  - Идентификатор используемой модели

- Есть ли метрики, характеризующие текущее состояние эксплуатационного сервера? Какие значения?
  - Метрики состояния сервера



# Насколько хорошо работает модель?



По [каким метрикам](#) можно оценить, что модель лучше предыдущей?

# Насколько хорошо работает модель?

- По **каким метрикам** можно оценить, что модель лучше предыдущей?
  - **метрики, характеризующие качество модели**, например:
    - MSE/MAE
    - AUC-ROC
    - Precision/Recall

# Насколько хорошо работает модель?

- По **каким метрикам** можно оценить, что модель лучше предыдущей?
  - **метрики, характеризующие качество модели**, например:
    - MSE/MAE
    - AUC-ROC
    - Precision/Recall
  
- Кто источник этих метрик?

# Насколько хорошо работает модель?

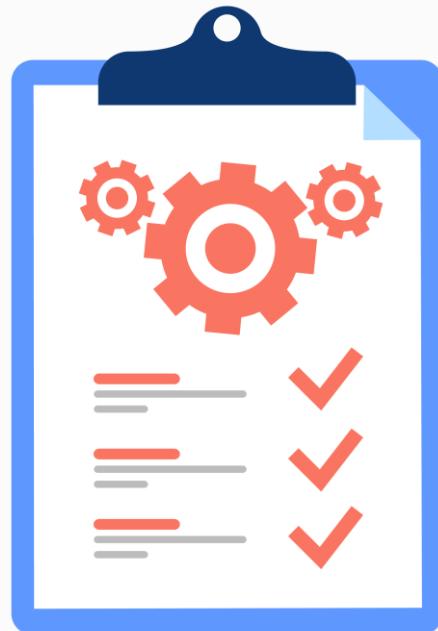
- По **каким метрикам** можно оценить, что модель лучше предыдущей?
  - **метрики, характеризующие качество модели**, например:
    - MSE/MAE
    - AUC-ROC
    - Precision/Recall
  
- Кто источник этих метрик?
  - Продуктовый менеджер, внутренние аналитики

# Каков процесс верификации модели?

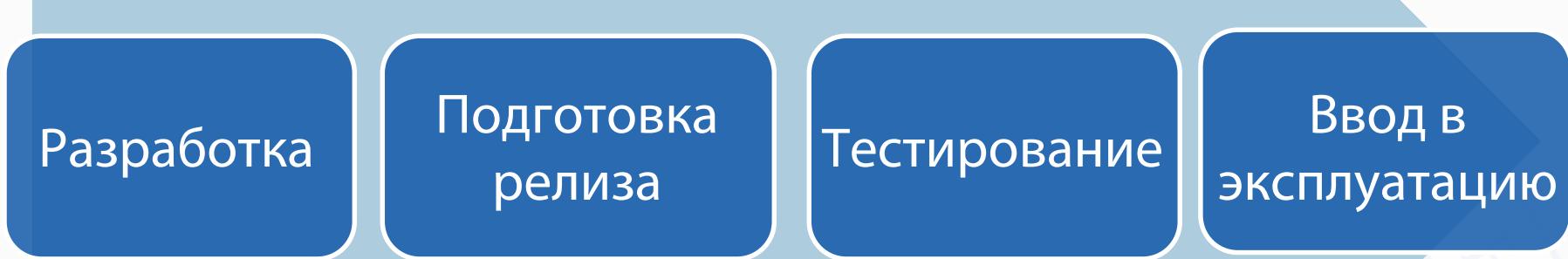
→ Как принимается решение о введении модели в эксплуатацию?

# Каков процесс верификации модели?

- Как принимается решение о введении модели в эксплуатацию?
- ✓ Нужен **процесс тестирования модели** и поэтапный ввод в эксплуатацию



# Поэтапный ввод в эксплуатацию



# Концепции процесса ввода в эксплуатацию

- Версионирование — каждый артефакт должен иметь идентификатор
- Воспроизводимость результата — артефакты с одинаковыми идентификаторами и параметрами в одной и той же среде производят одинаковый результат



# Версионирование моделей



Каждая обученная модель должна иметь идентификатор, используемый на всех этапах внедрения

# Версионирование моделей



Каждая обученная модель должна иметь идентификатор, используемый на всех этапах внедрения



Каждая обученная модель связана с параметрами, с которыми она обучена, и метриками качества

# Версионирование моделей



Каждая обученная модель должна иметь идентификатор, используемый на всех этапах внедрения



Каждая обученная модель связана с параметрами, с которыми она обучена, и метриками качества



Версия модели является идентификатором и ассоциируется с параметрами и метриками

# Версионирование моделей



Каждая обученная модель должна иметь идентификатор, используемый на всех этапах внедрения



Каждая обученная модель связана с параметрами, с которыми она обучена, и метриками качества



Версия модели является идентификатором и ассоциируется с параметрами и метриками

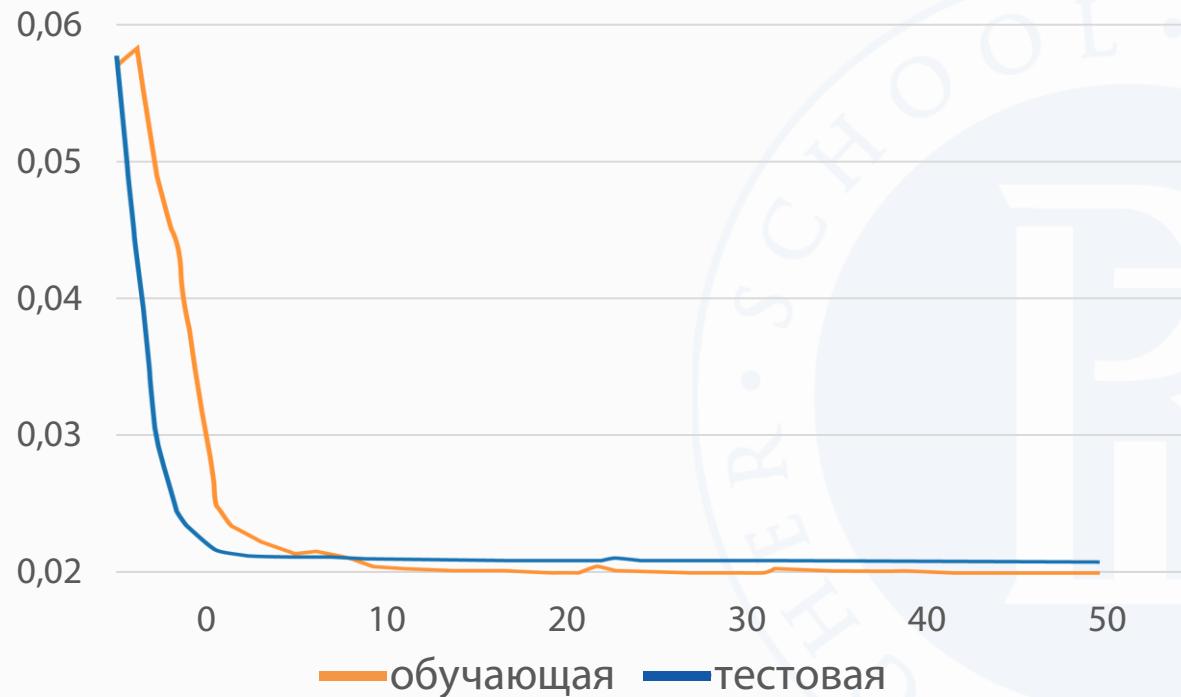


Все версионированные артефакты удобнее хранить в одном месте (реестре)

# Метрики

→ Значения ключевых метрик характеризуют модель и используются как результаты ее верификации

→ Например, значения целевой функции на тестовой выборке



# Аналогии с внедрением кода

Выкладка новой версии сервиса схожа с внедрением новой или обновленной модели:

- Docker image аналогичен обученной модели



# Аналогии с внедрением кода

Выкладка новой версии сервиса схожа с внедрением новой или обновленной модели:

- Docker image аналогичен обученной модели
- Поэтапная выкладка кода или модели с последующим тестированием и принятием решения о продвижении на следующий этап

# Аналогии с внедрением кода

Выкладка новой версии сервиса схожа с внедрением новой или обновленной модели:

- Docker image аналогичен обученной модели
- Поэтапная выкладка кода или модели с последующим тестированием и принятием решения о продвижении на следующий этап
- Метрики для отображения состояния сервиса или работы модели

# Резюме



Всё, что внедряется в эксплуатацию (код или модель),  
должно иметь: мониторинг, метрики, процесс поэтапного  
внедрения с верификацией



Для реализации требований нужны инструменты  
оперирующие с версиями артефактов  
и поддерживающие воспроизводимость результатов

# Резюме



Всё, что внедряется в эксплуатацию (код или модель),  
должно иметь: мониторинг, метрики, процесс поэтапного  
внедрения с верификацией



Для реализации требований нужны инструменты  
оперирующие с версиями артефактов  
и поддерживающие воспроизводимость результатов



Далее: организация воспроизводимого обучения моделей

# **Воспроизводимое обучение моделей**



# План



Процесс разработки моделей



Архитектура ML-платформы



Особенности работы с большими данными

# Процесс разработки моделей



Постановка задачи, выбор целевой функции и значений ключевых метрик



Итеративное экспериментирование с параметрами и алгоритмами обучения модели



Подготовка модели к дистрибуции



Введение успешной модели в эксплуатацию

# Процесс разработки моделей



Постановка задачи, выбор целевой функции и значений ключевых метрик



Итеративное экспериментирование с параметрами и алгоритмами обучения модели



Подготовка модели к дистрибуции



Введение успешной модели в эксплуатацию

Что нужно учесть при построении решения?

# Характеристики процесса

- Интерактивная среда, например, созданная в Jupyter Notebook, хаотично меняется во время экспериментов



# Характеристики процесса

Интерактивная среда, например, созданная в Jupyter Notebook, хаотично меняется во время экспериментов



Набор параметров окружения сложно сохранить для повторного воспроизведения эксперимента

**Restart kernel and re-run the whole notebook?** ×

Are you sure you want to restart the current kernel and re-execute the whole notebook? All variables and outputs will be lost.

Continue running Restart & run all cells

# Характеристики процесса

Интерактивная среда, например, созданная в Jupyter Notebook, хаотично меняется во время экспериментов

Набор параметров окружения сложно сохранить для повторного воспроизведения эксперимента

- Нет стандартного формата хранения и дистрибуции моделей, связь между моделью и параметрами среды не всегда сохраняется



TensorFlow



Keras

# Характеристики процесса

Интерактивная среда, например, созданная в Jupyter Notebook, хаотично меняется во время экспериментов

Набор параметров окружения сложно сохранить для повторного воспроизведения эксперимента

Нет стандартного формата хранения и дистрибуции моделей, связь между моделью и параметрами среды не всегда сохраняется

- Затруднено создание централизованного сервиса хранения и версионирования моделей

# Архитектура ML-платформы

- Предоставляет каркас для построения процесса
- Интегрируется с другими инструментами и сервисами
- Веб-интерфейс

ML Проект  
Управление  
экспериментами  
и внедрением



# Архитектура ML-платформы

- Предоставляет каркас для построения процесса
- Интегрируется с другими инструментами и сервисами
- Веб-интерфейс



# Архитектура ML-платформы

- Предоставляет каркас для построения процесса
- Интегрируется с другими инструментами и сервисами
- Веб-интерфейс



# Примеры ML-платформ



Azure ML

mlflow



Amazon SageMaker



Google DataLab

# Работа с экспериментами: задача

- Контроль запусков всех экспериментов с сохранением параметров
- Для воспроизведения в изолированной среде с сохранением метрик и артефактов, произведенных во время работы

# ML-проекты: задачи

- Связь моделей и экспериментов
- Обеспечение воспроизводимости результата в изолированной среде
- Определение точек входа проекта
- Аналогия: Dockerfile

# Модели

ML-библиотеки не имеют единого формата, как и среды интеграций моделей

- Нужна дополнительная разработка на каждую пару модели и среды интеграции

Реестр моделей [сериализует](#) артефакты моделей, устранивая необходимость дополнительной разработки

# Реестр моделей: структура модели

При сохранении модели сохраняются:

- параметры, метаинформация о модели
  - версии зависимостей, точки входа / API
- артефакты самих моделей

# Реестр моделей: структура модели

При сохранении модели сохраняются:

- параметры, метаинформация о модели
  - версии зависимостей, точки входа / API

- артефакты самих моделей



Pickle

для моделей описанных на Python  
(например, Scikit learn)



SavedModel

стандартный формат модели TensorFlow



SparkContext

контекст для Spark MLlib

# Реестр моделей: задачи

- Унификация интерфейсов моделей, использующих разные библиотеки
- Интеграция модели в сервис, конвейер данных, другую среду эксплуатации (с помощью API)
- Описание зависимостей для воспроизводимости построения модели
- Инструменты для упрощения создания моделей

# Реестр моделей: задачи

- Централизованное хранение версий моделей для удобного поиска
- Информация о том, какая модель используется на какой среде
- История всех версий и их использования в средах
- Аналогия: Dockerhub + Continuous Integration service

# Работа с большими данными

ML-платформа координирует процесс  
независимо от инфраструктуры

- Распределенный запуск на кластере
  - обучение моделей в Spark

# Работа с большими данными

ML-платформа координирует процесс  
независимо от инфраструктуры

→ Распределенный запуск на кластере

- обучение моделей в Spark

→ Параллельный запуск экспериментов

- online и offline
- подбор гиперпараметров

# Работа с большими данными

ML-платформа координирует процесс независимо от инфраструктуры

→ Распределенный запуск на кластере

- обучение моделей в Spark

→ Параллельный запуск экспериментов

- online и offline
- подбор гиперпараметров

→ Облачные хранилища

- интеграция с AWS
- хранение артефактов: удаленный реестр моделей, docker-образов

# ML-платформа: основные пользователи

## → Команда аналитиков

- база экспериментов вносит порядок благодаря фиксации параметров и результатов

# ML-платформа: основные пользователи

## → Команда аналитиков

- база экспериментов вносит порядок благодаря фиксации параметров и результатов

## → Релиз-инженеры и DevOps

- реестр — проще работать с унифицированными API

# ML-платформа: основные пользователи

## → Команда аналитиков

- база экспериментов вносит порядок благодаря фиксации параметров и результатов

## → Релиз-инженеры и DevOps

- реестр — проще работать с унифицированными API

## → Разработчики и интеграторы ML-решений

- универсальный интерфейс моделей

# ML-платформа: основные пользователи



## Команда аналитиков

- база экспериментов вносит порядок благодаря фиксации параметров и результатов



## Релиз-инженеры и DevOps

- реестр — проще работать с унифицированными API



## Разработчики и интеграторы ML-решений

- универсальный интерфейс моделей



## Разработчики собственных ML-библиотек

- быстрый старт использования модели в цикле разработки

# Резюме



Процесс разработки моделей включает множество этапов помимо экспериментов



ML-платформы обеспечивают удобную и эффективную разработку моделей

# Резюме



Процесс разработки моделей включает **множество** этапов помимо экспериментов

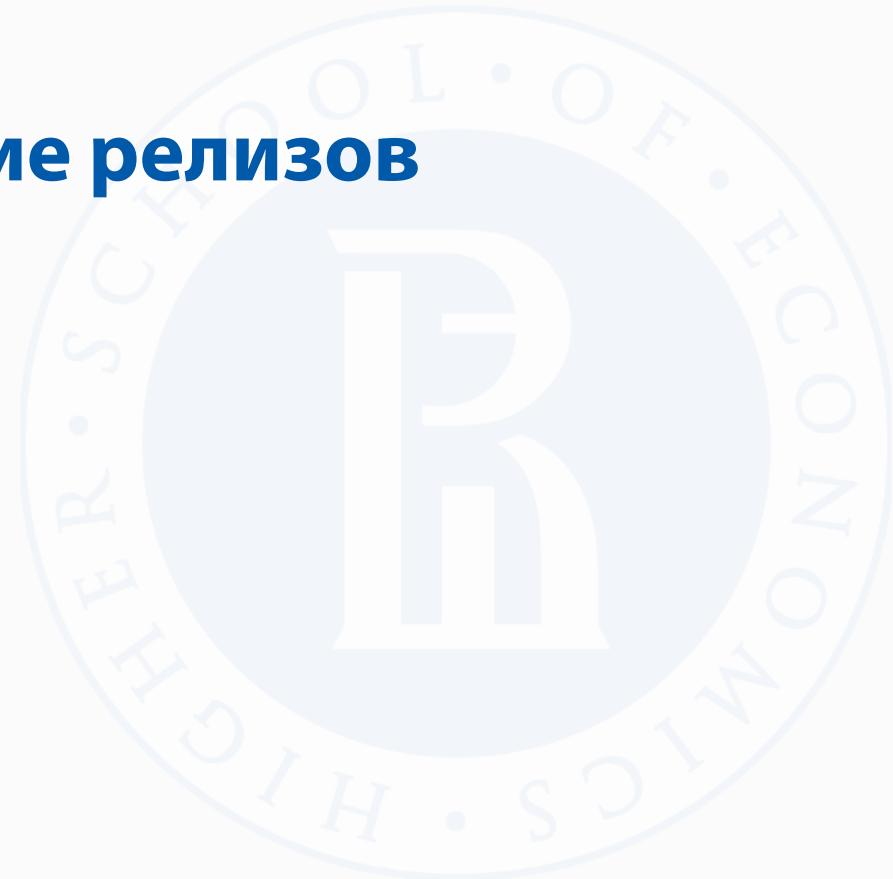


**ML-платформы** обеспечивают удобную и эффективную разработку моделей



Далее: планирование и ввод в эксплуатацию обученных моделей

# **Версионирование релизов**



# План



Компоненты выкладки модели



Процесс выкладки модели



# Компоненты выкладки модели



Проект ML-сервиса



Эксперимент, ассоциированный с проектом



Модель, готовая к дистрибуции



Введение модели в эксплуатацию

# ML-проект

→ Проект — описание среды экспериментов

Библиотеки  
Переменные



# ML-проект

→ Проект — описание среды экспериментов

Библиотеки  
Переменные

→ Текстовые файлы проекта следует хранить в системе  
контроля версий

Библиотеки  
Переменные

hash1



Библиотеки  
Переменные

hash2

# ML-проект

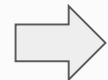
→ Проект — описание среды экспериментов

Библиотеки  
Переменные

→ Текстовые файлы проекта следует хранить в системе контроля версий

Библиотеки  
Переменные

hash1



Библиотеки  
Переменные

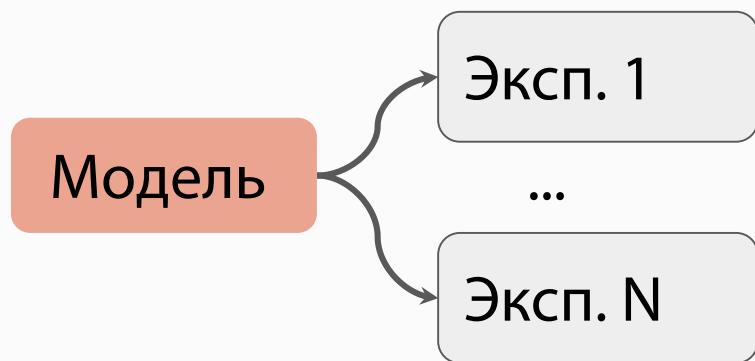
hash2

→ Версия проекта ассоциирована с коммитом

- информация отображается в ML-платформе

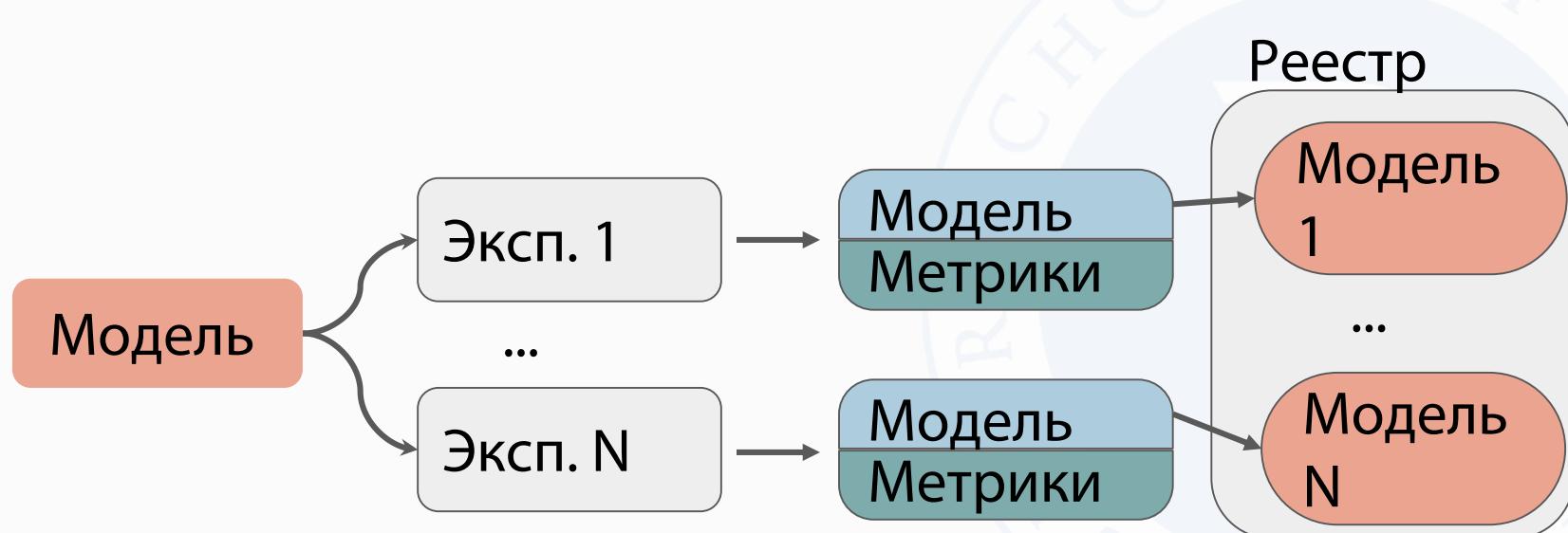
# Эксперименты

→ Каждый запуск обладает собственным идентификатором



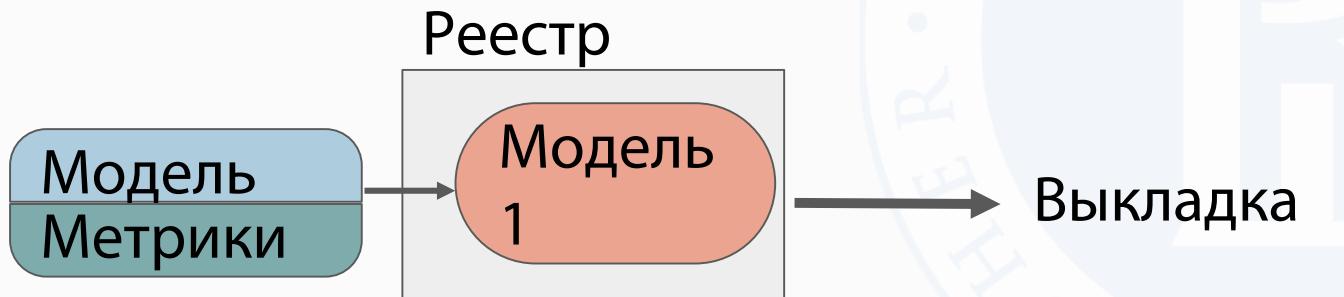
# Эксперименты

- Каждый запуск обладает собственным идентификатором
- Каждый запуск эксперимента сохраняет артефакт в реестр моделей



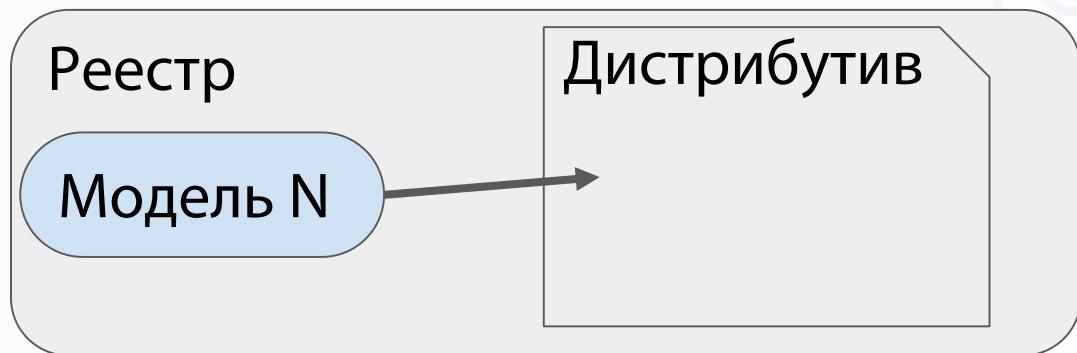
# Эксперименты

- Каждый запуск обладает собственным идентификатором
- Каждый запуск эксперимента сохраняет артефакт в реестр моделей
- Идентификатор используется в других процессах
  - при выкладке для получения артефактов из реестра



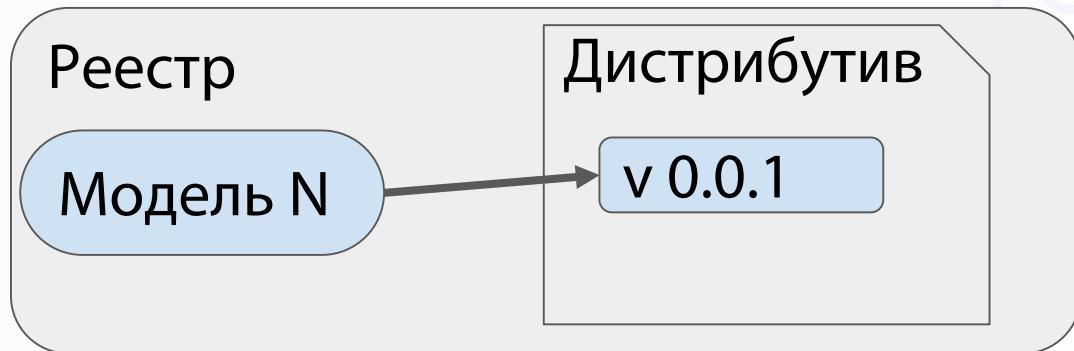
# Поэтапная выкладка модели

→ Из модели собирается дистрибутив



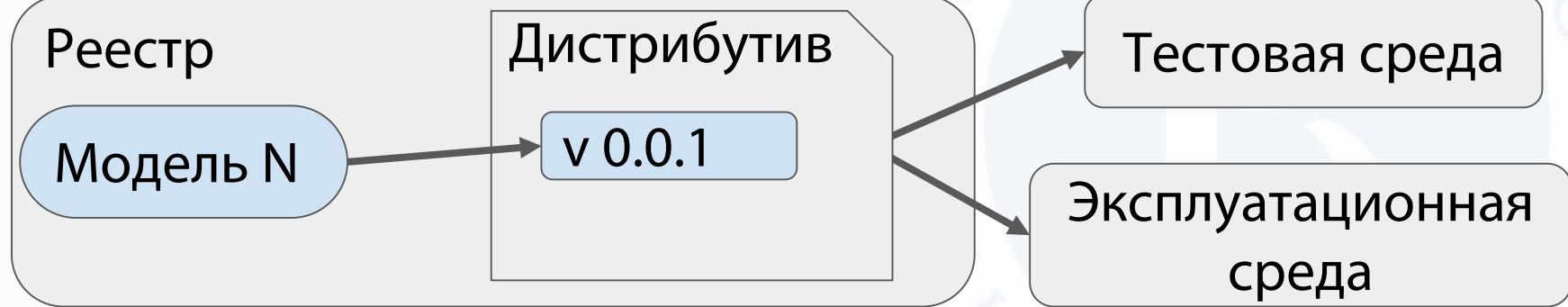
# Поэтапная выкладка модели

- Из модели собирается дистрибутив
- Дистрибутив версионируется



# Поэтапная выкладка модели

- Из модели собирается дистрибутив
- Дистрибутив версионируется
- Дистрибутив можно выкладывать в среды назначения



# Поэтапная выкладка модели

- Из модели собирается дистрибутив
- Дистрибутив версионируется
- Дистрибутив можно выкладывать в среды назначения
- Выпуски дистрибутива будут иметь **инкрементальные** версии, прошлые версии **сохраняются** в реестре

## Реестр

Дистрибутив

v 0.0.1

Дистрибутив

v 0.0.2

# Подготовка версии модели к дистрибуции

Сборка модели для дистрибуции с помощью реестра

Дистрибутив содержит:

- Модель — артефакт эксперимента
- Описание API
- Метаинформацию
  - название
  - версия
  - дата создания



# Воспроизводимость и версионирование



Точкой фиксации параметров и результатов является запуск эксперимента



Проект и модель отвечают за окружение и интерфейс, метаданные сохраняются в системе контроля версий



Запуск эксперимента сохраняет изолированную обученную модель в реестре в указанном формате

# Форматы версии



Одновременно может проходить много экспериментов в распределенной системе, поэтому хэш удобнее для идентификации эксперимента



Зарегистрированные модели, как правило, используют инкрементальное версионирование для понимания, какая из двух версий более новая

# Резюме



Правильно определенные компоненты выкладки помогают отследить **весь процесс разработки** модели и **унифицируют** процесс ввода модели в эксплуатацию



Процесс ввода в эксплуатацию **снижает риски** неработоспособности сервера модели после обновления

# Резюме



Правильно определенные компоненты выкладки помогают отследить **весь процесс разработки** модели и **унифицируют** процесс ввода модели в эксплуатацию



Процесс ввода в эксплуатацию **снижает риски** неработоспособности сервера модели после обновления



Далее:

- Практическая работа по разработке и введению модели в эксплуатацию
- Приемка изменений и откат некорректных изменений

# **Прием изменений и ввод в эксплуатацию, техническая сторона**



# План



Этапы внедрения изменений



Откат изменений в случае сбоев



Обратная совместимость изменений



Доступность сервиса во время выкладки изменений

# Этапы внедрения изменений

Эксплуатационная среда

Исходное  
состояние

Тестовая среда

Дистрибутив

Время

# Этапы внедрения изменений

Эксплуатационная среда

Исходное  
состояние

Тестовая среда

Время

Дистрибутив → Верификация

# Этапы внедрения изменений

Эксплуатационная среда

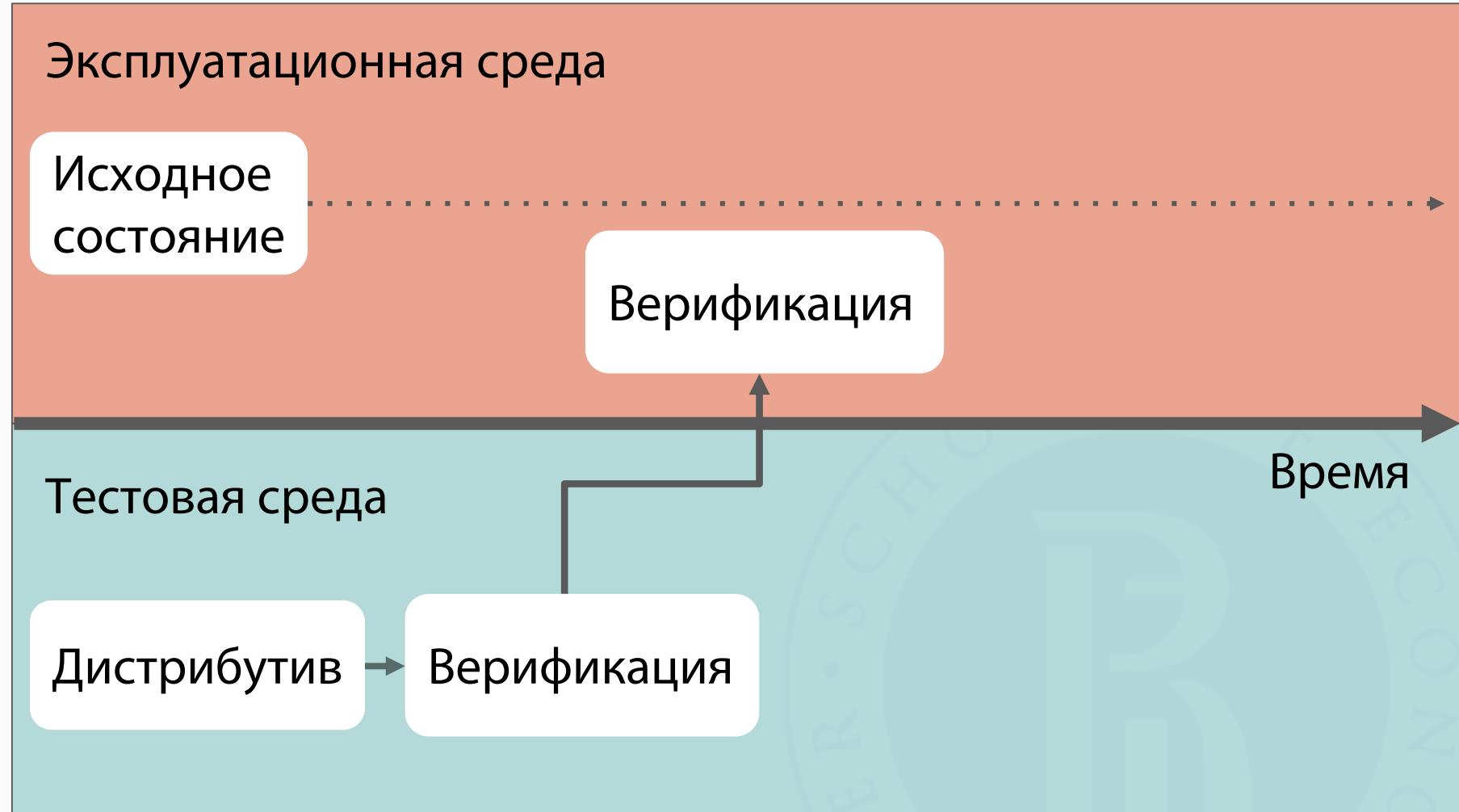
Исходное  
состояние

Тестовая среда

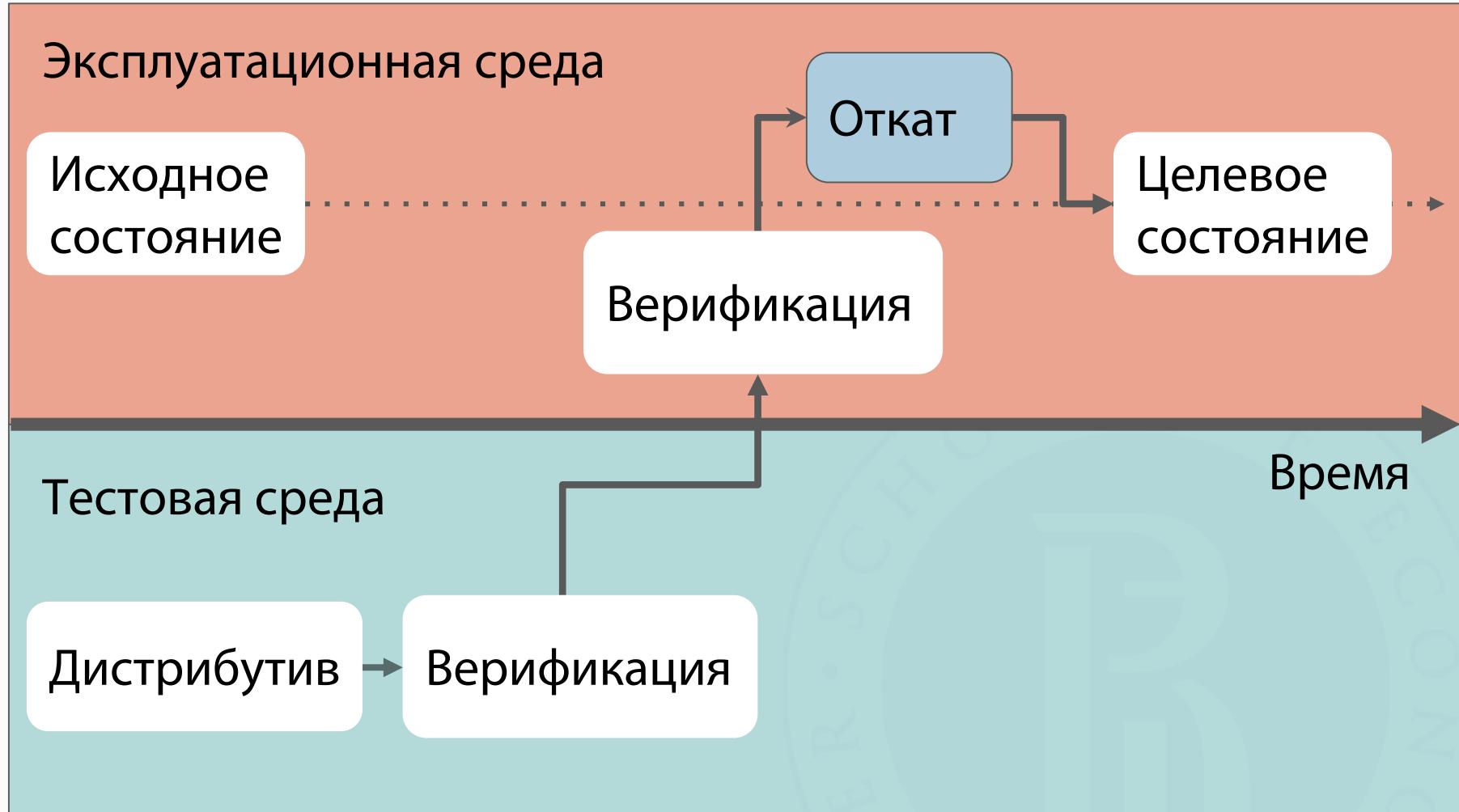
Время



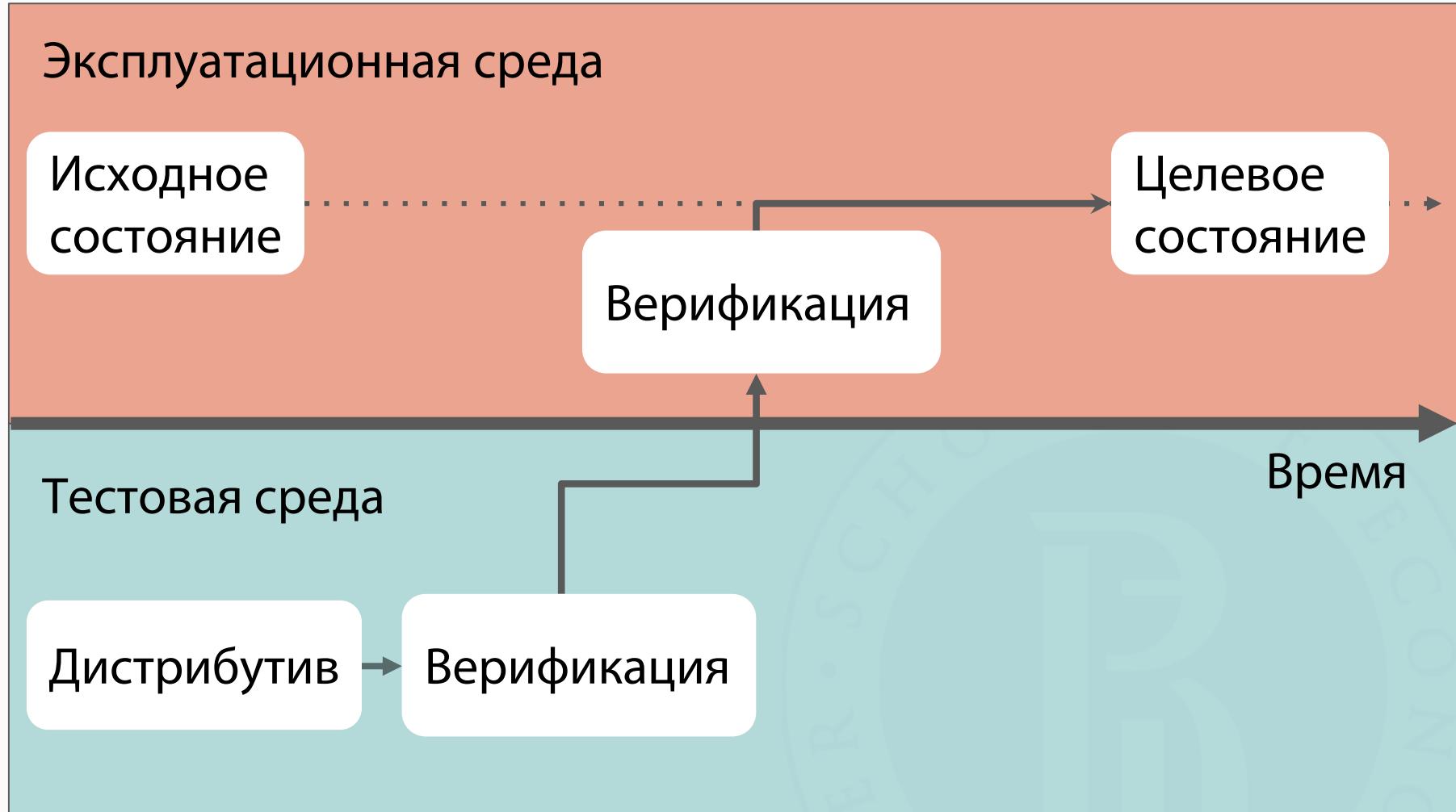
# Этапы внедрения изменений



# Этапы внедрения изменений



# Этапы внедрения изменений



# Приемка внедрений в эксплуатацию

→ Успешность выкладки верифицируется по показателям состояния сервиса

- Health-check endpoint

- URL сервиса, сообщает состояние сервиса

- Метрики использования CPU/RAM



# Верификация состояния сервиса

- Использование пограничных значений для показателей:
- Не отвечает Health-check
  - Слишком большой рост потребления CPU



- Пользователи сообщают об ошибках и неожиданных ответах от сервиса

# Действия в случае сбоев

- Идентифицировать сбоящий компонент
- Наиболее частое решение проблемы — выкладка предыдущей стабильной версии сбоящего компонента



# Откат на предыдущую версию



Откат — повторный ввод в эксплуатацию стабильной предыдущей версии

# Откат на предыдущую версию

- Откат — повторный ввод в эксплуатацию стабильной предыдущей версии
- Если предыдущая стабильная версия также [не работает](#):

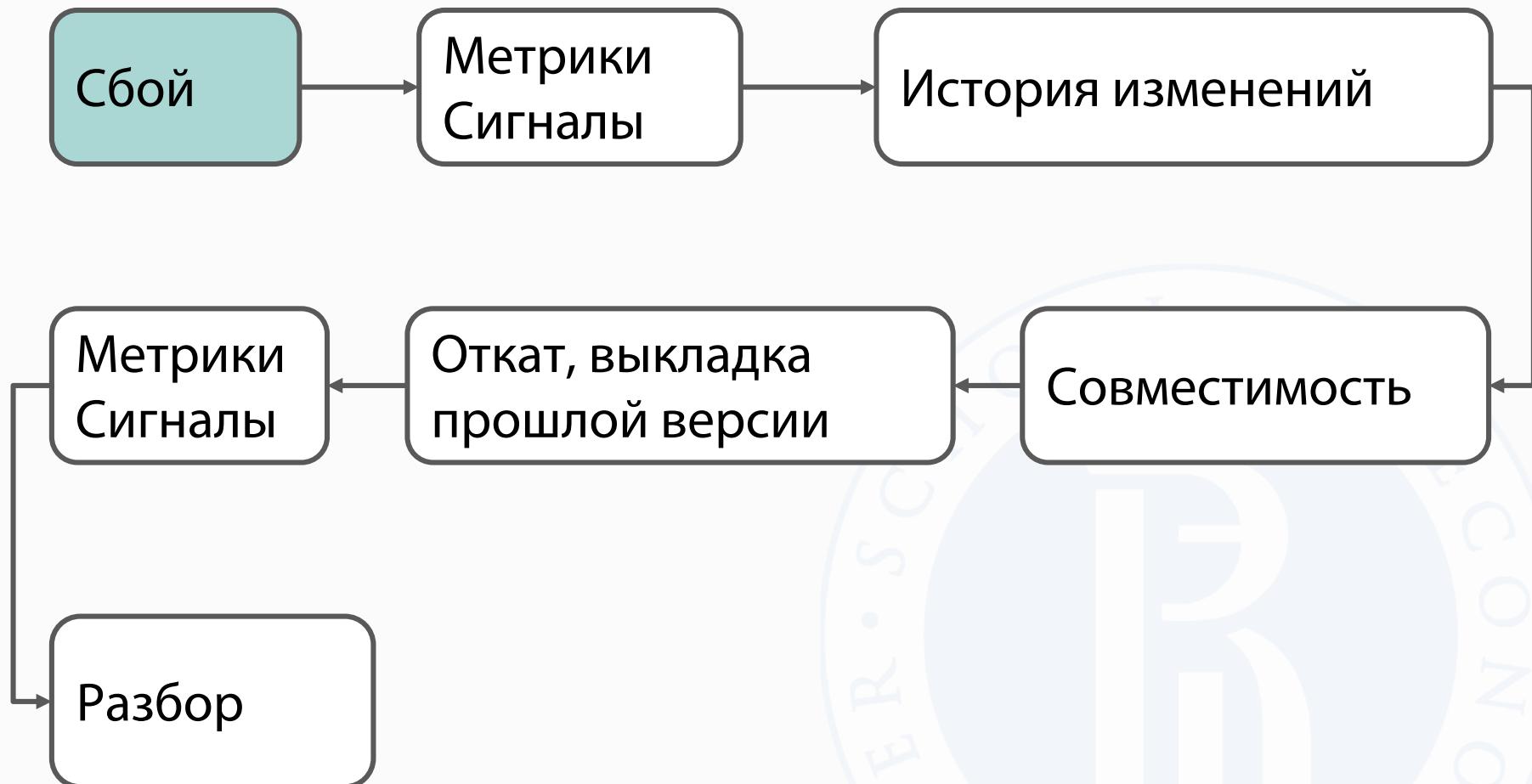
# Откат на предыдущую версию

- Откат — повторный ввод в эксплуатацию стабильной предыдущей версии
- Если предыдущая стабильная версия также **не работает**:
  - проверить изменение окружения

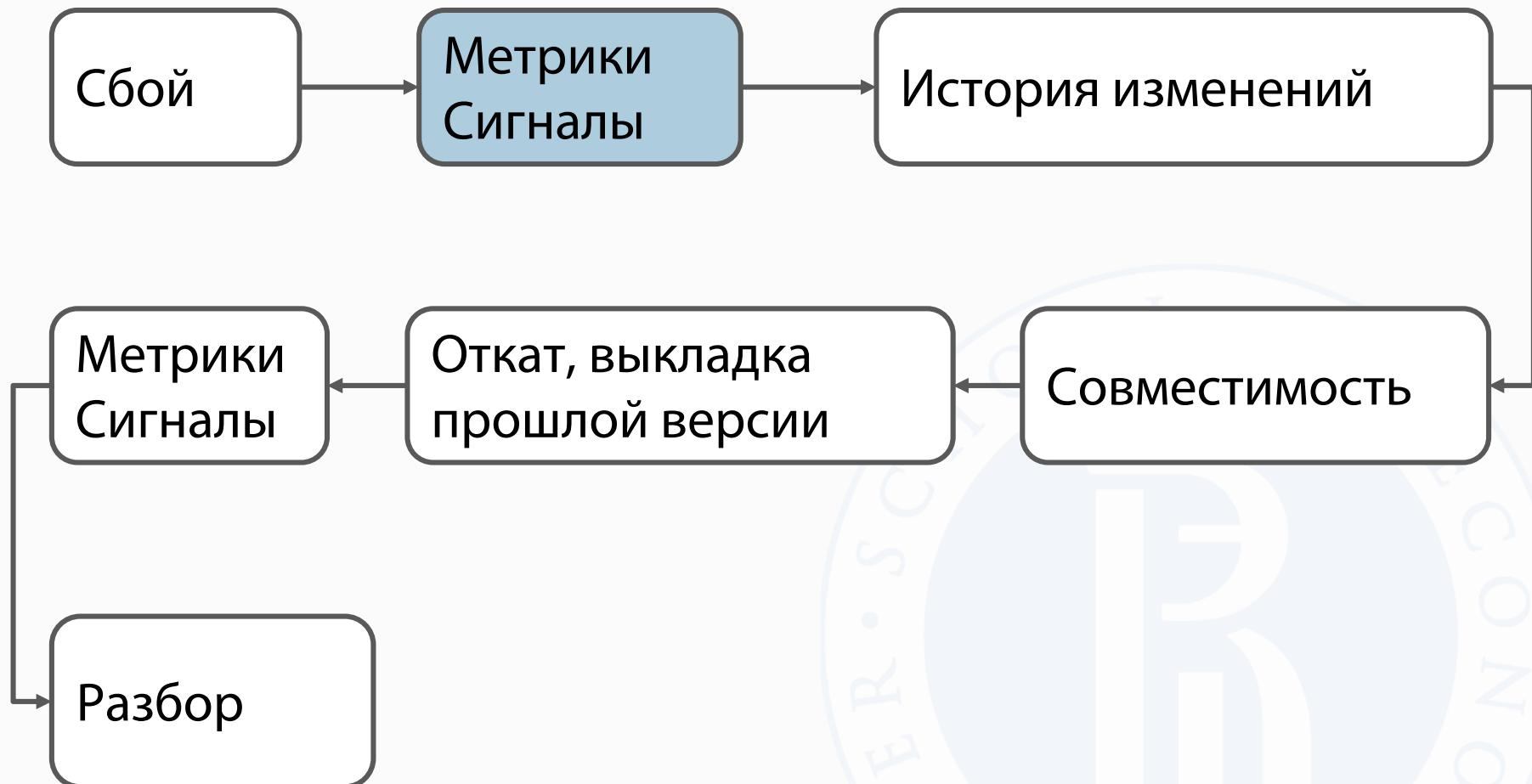
# Откат на предыдущую версию

- Откат — повторный ввод в эксплуатацию стабильной предыдущей версии
- Если предыдущая стабильная версия также **не работает**:
  - проверить изменение окружения
  - прочие изолированные от версии модели настройки

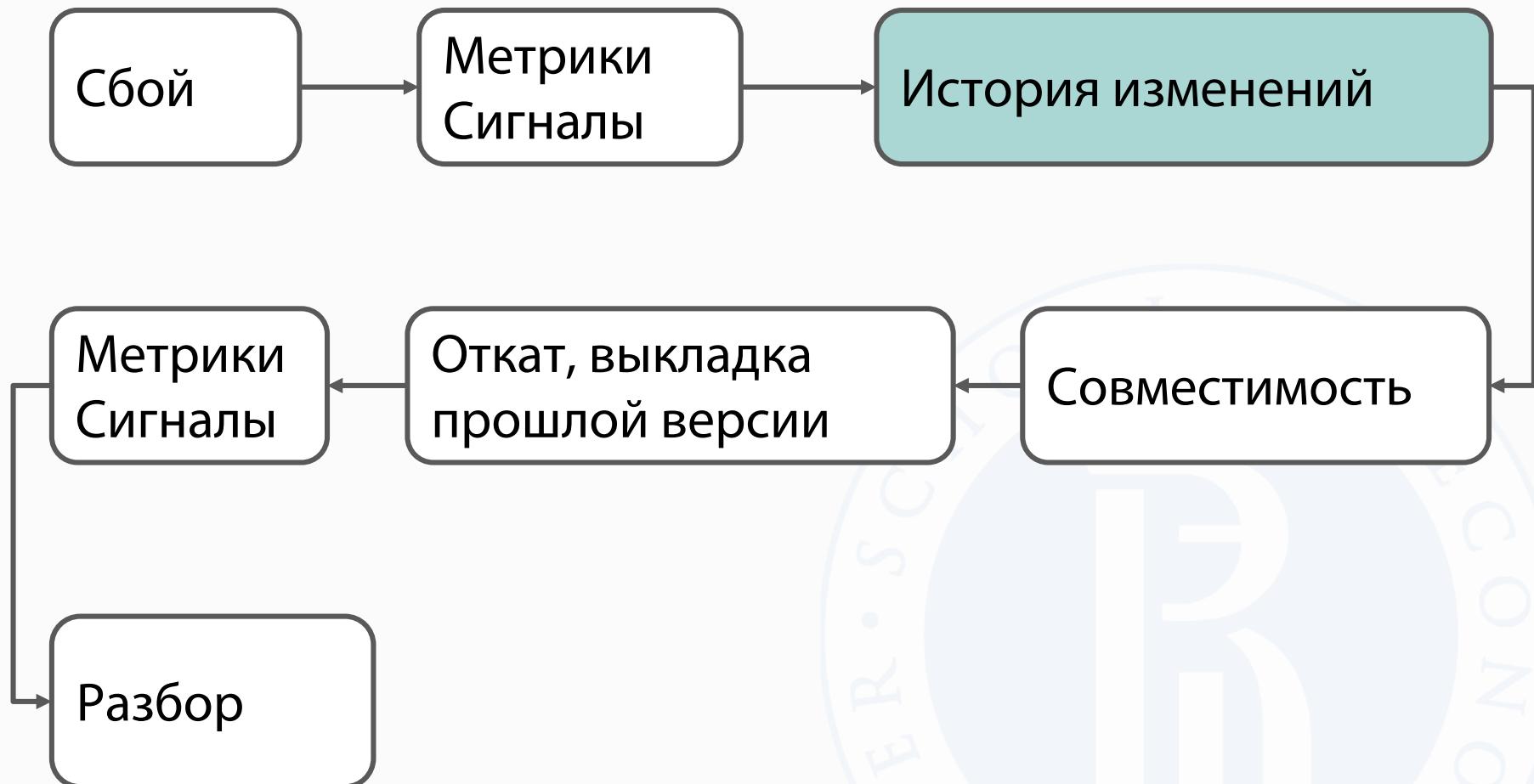
# Алгоритм принятия решений об откате



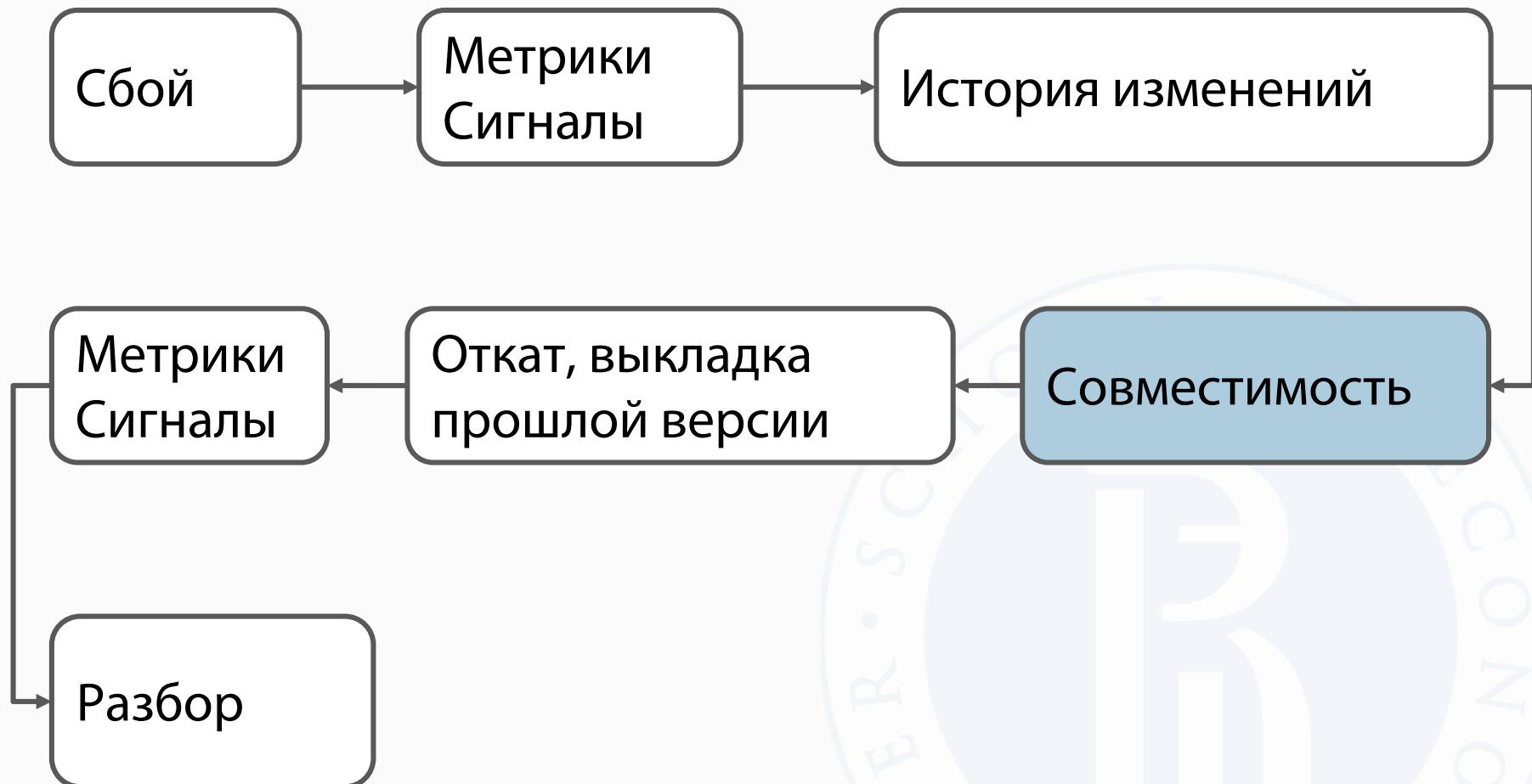
# Алгоритм принятия решений об откате



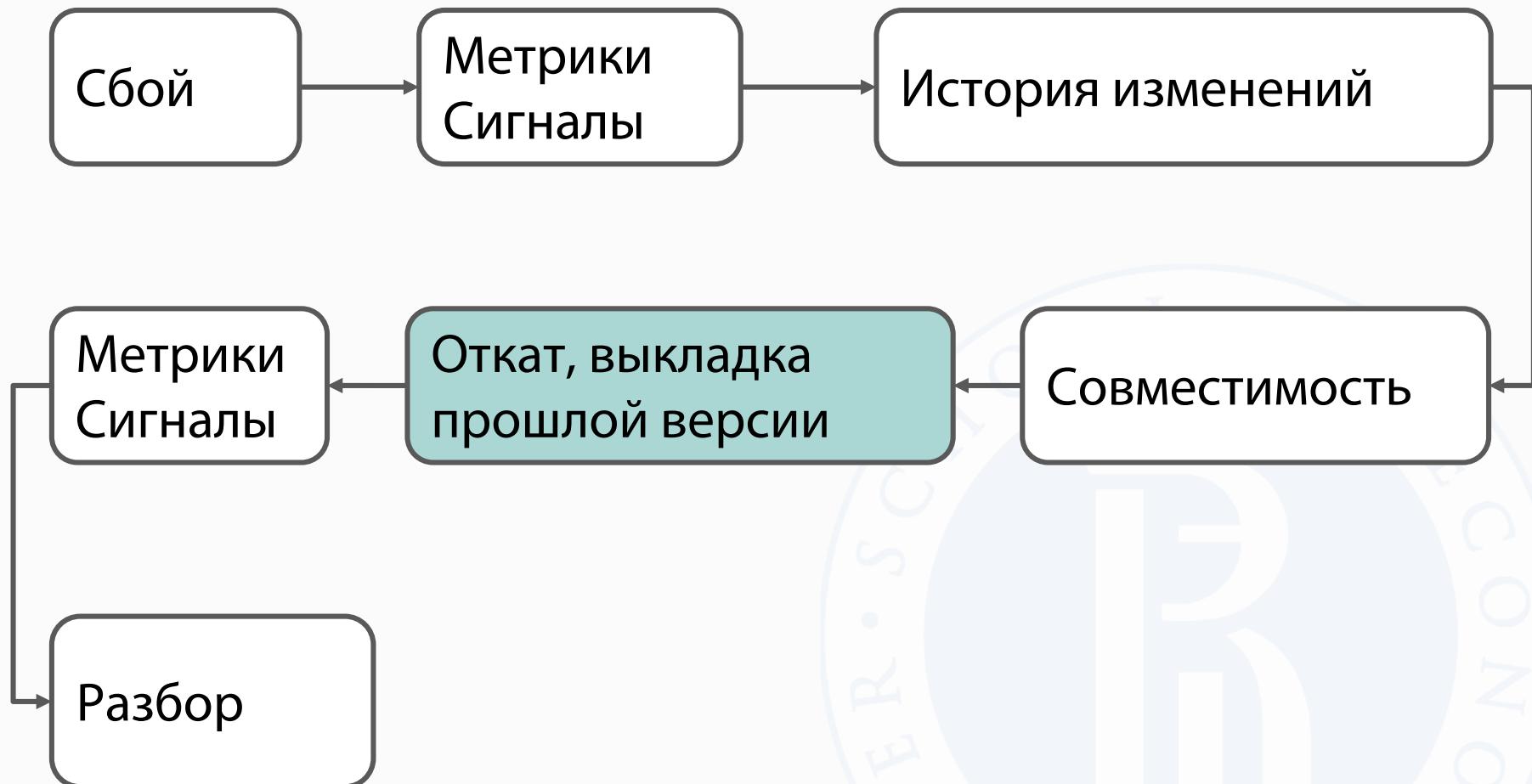
# Алгоритм принятия решений об откате



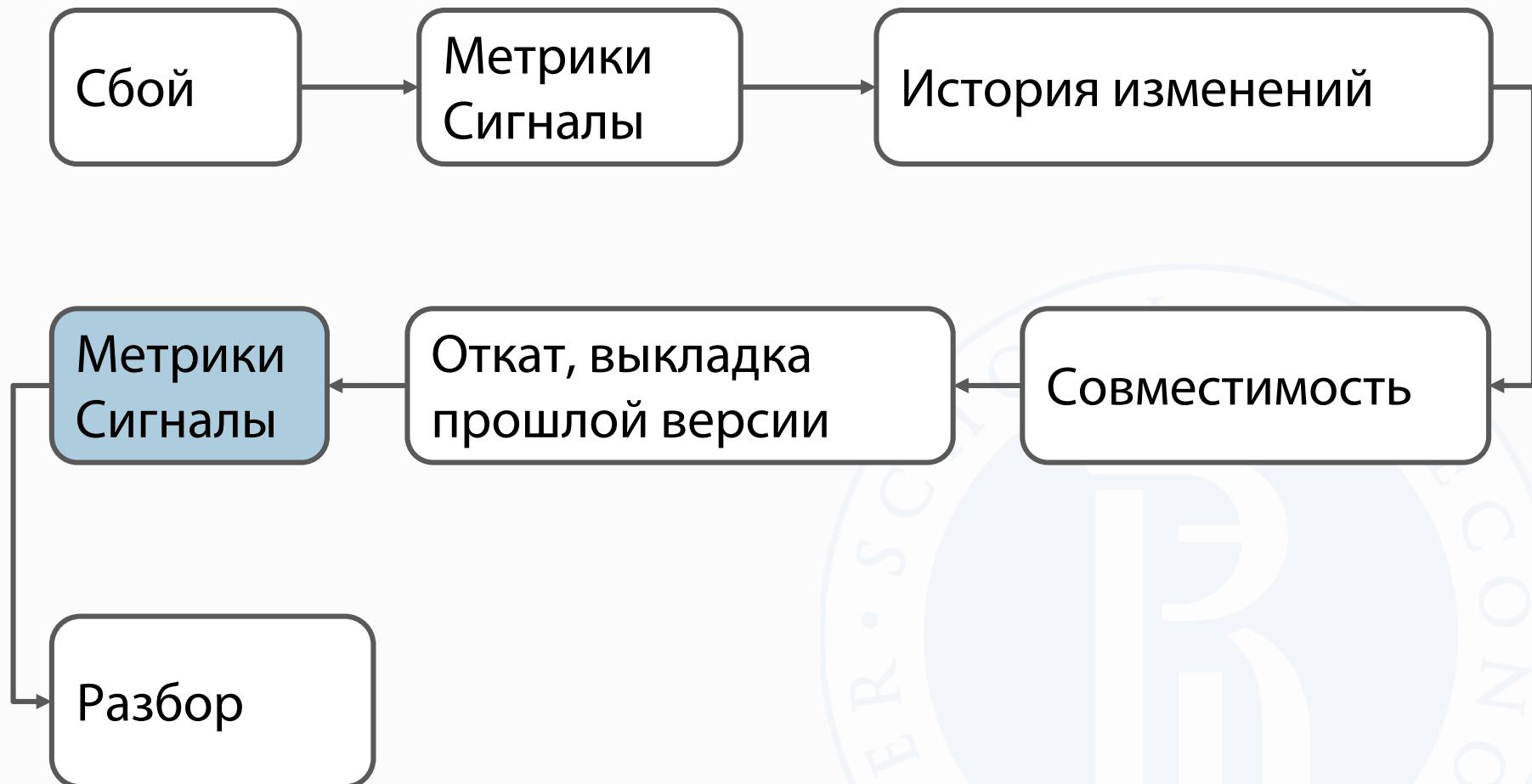
# Алгоритм принятия решений об откате



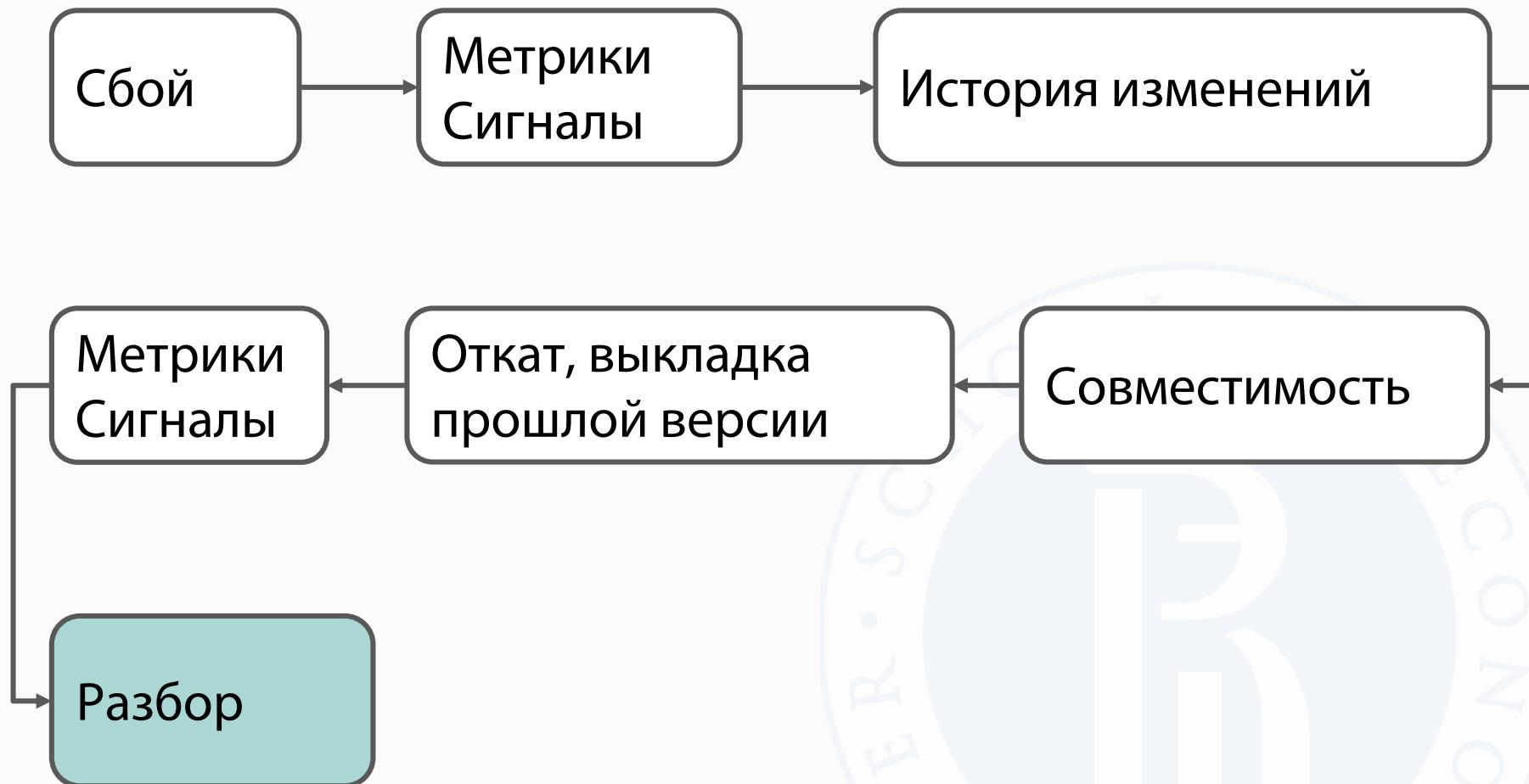
# Алгоритм принятия решений об откате



# Алгоритм принятия решений об откате



# Алгоритм принятия решений об откате

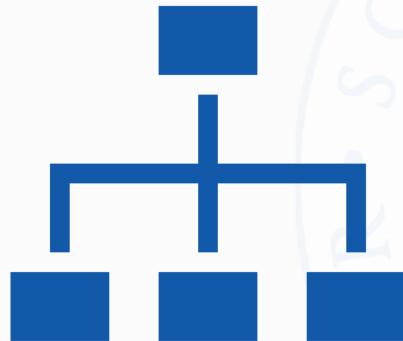


# Обратная совместимость

- Изменение интерфейса сервиса может сделать новую версию несовместимой с предыдущими
- Часто поддержка обратной совместимости приводит к усложнению кода
- Несовместимые изменения одного из компонентов могут привести к цепочке изменений других компонентов и усложнению введения в эксплуатацию

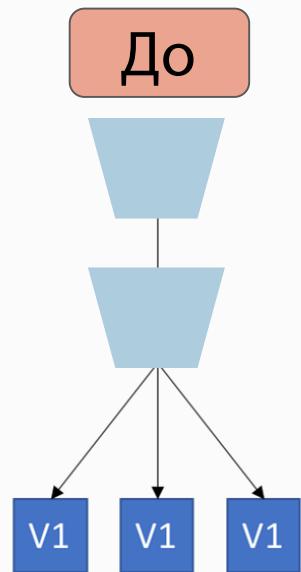
# Доступность сервиса в процессе релиза

- Старт сервиса на новой машине может занимать некоторое время
- Для постоянного доступа необходимо иметь по крайней мере одну машину с работающим сервисом



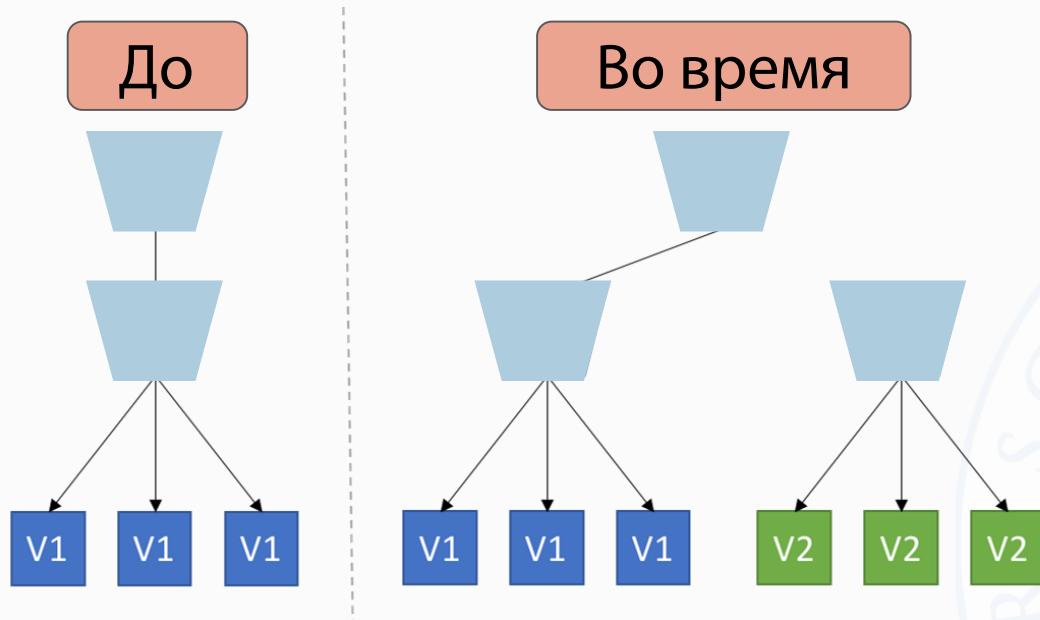
# Blue-Green Deployment

Основная идея: мгновенное переключение между старой и новой версиями



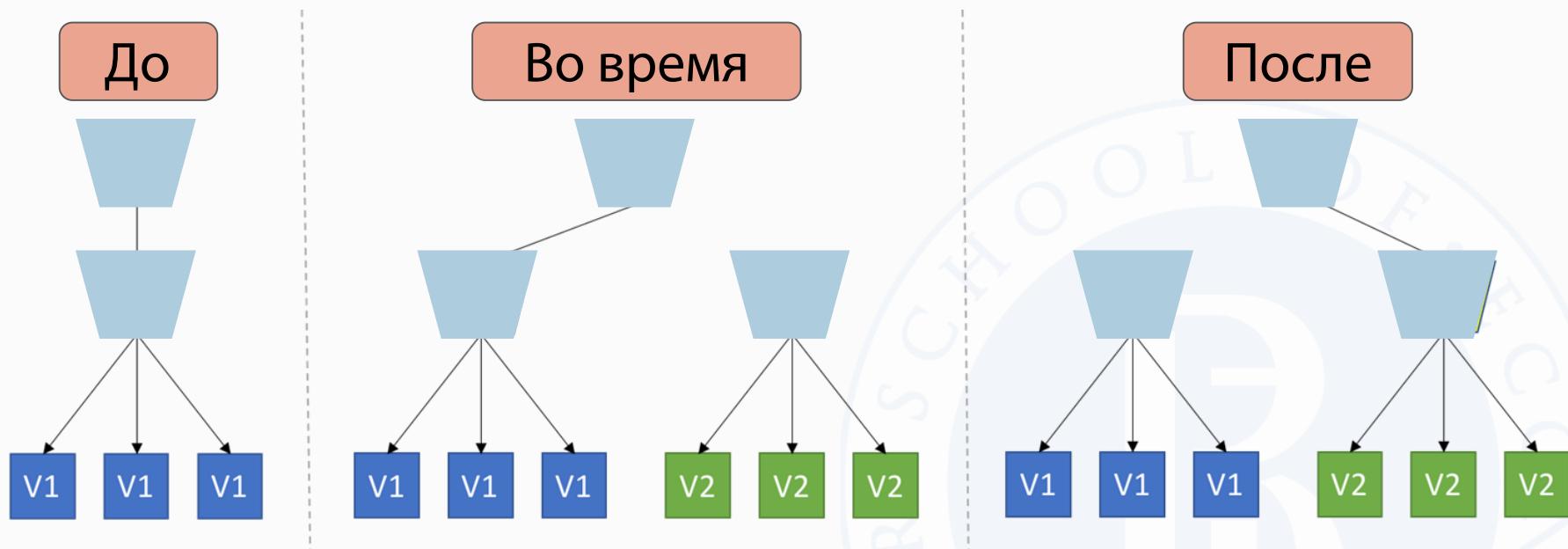
# Blue-Green Deployment

Основная идея: мгновенное переключение между старой и новой версиями



# Blue-Green Deployment

Основная идея: мгновенное переключение между старой и новой версиями



# Этапы внедрения в эксплуатацию



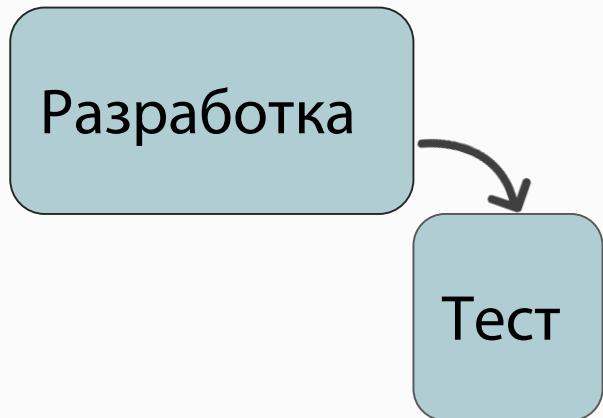
Полный цикл внедрения может включать этапы:

Разработка



# Этапы внедрения в эксплуатацию

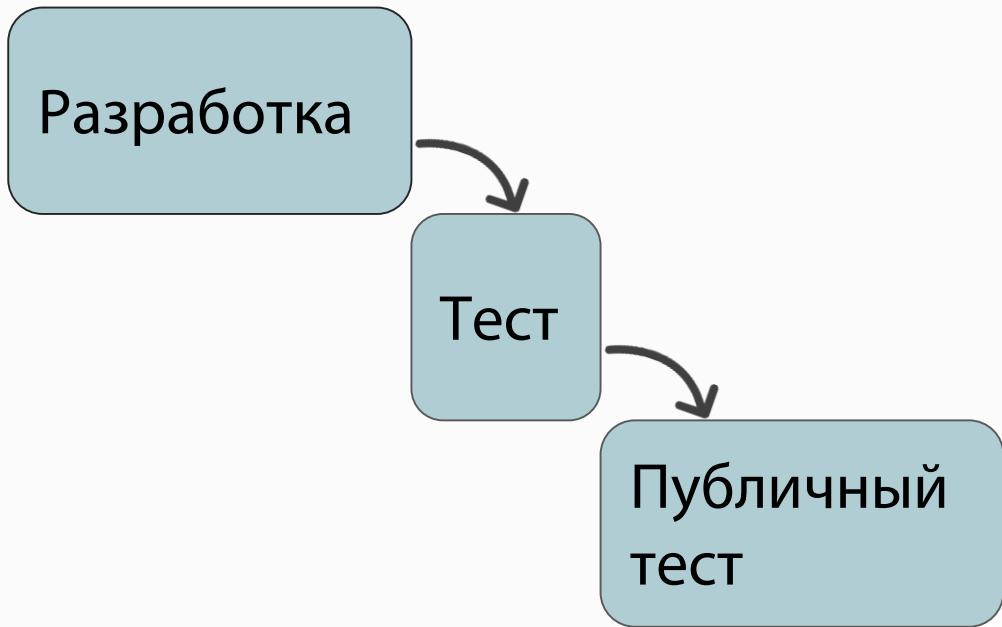
→ Полный цикл внедрения может включать этапы:



# Этапы внедрения в эксплуатацию

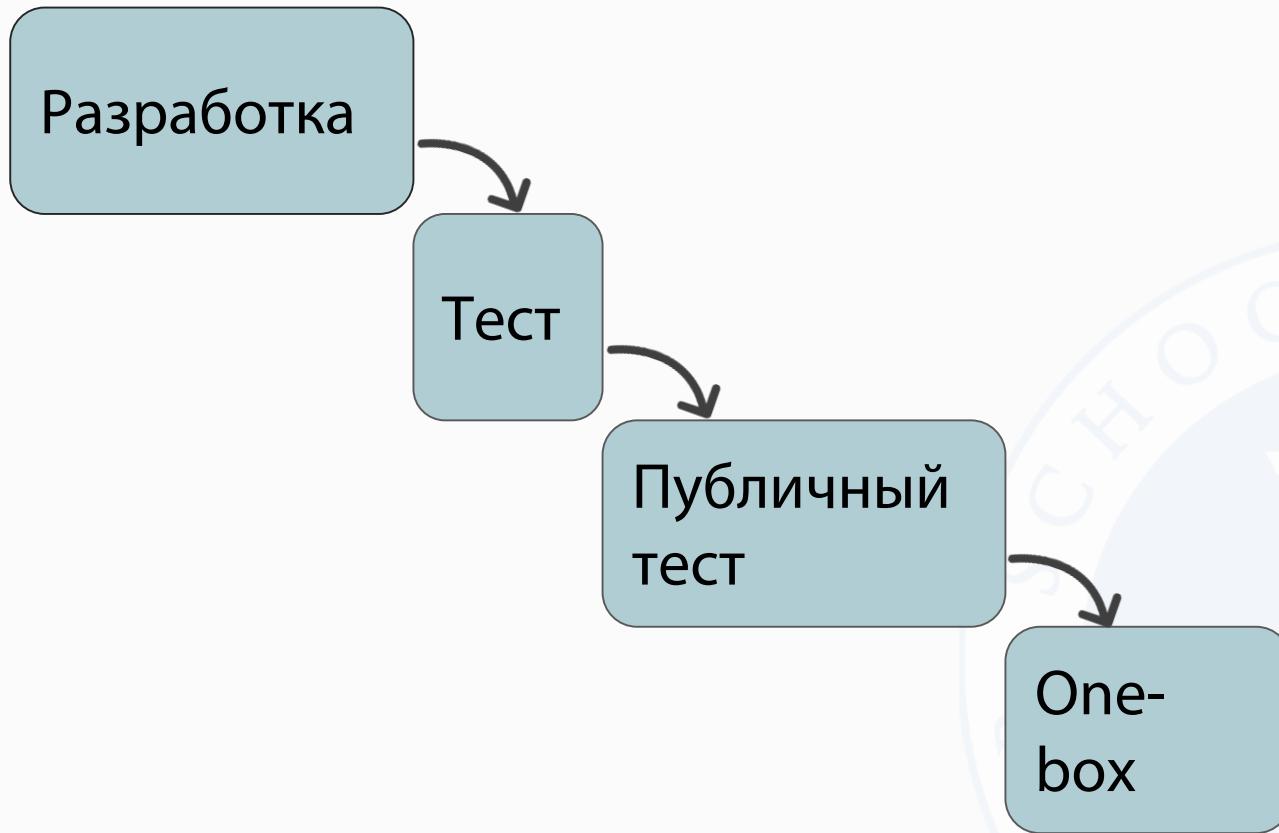


Полный цикл внедрения может включать этапы:



# Этапы внедрения в эксплуатацию

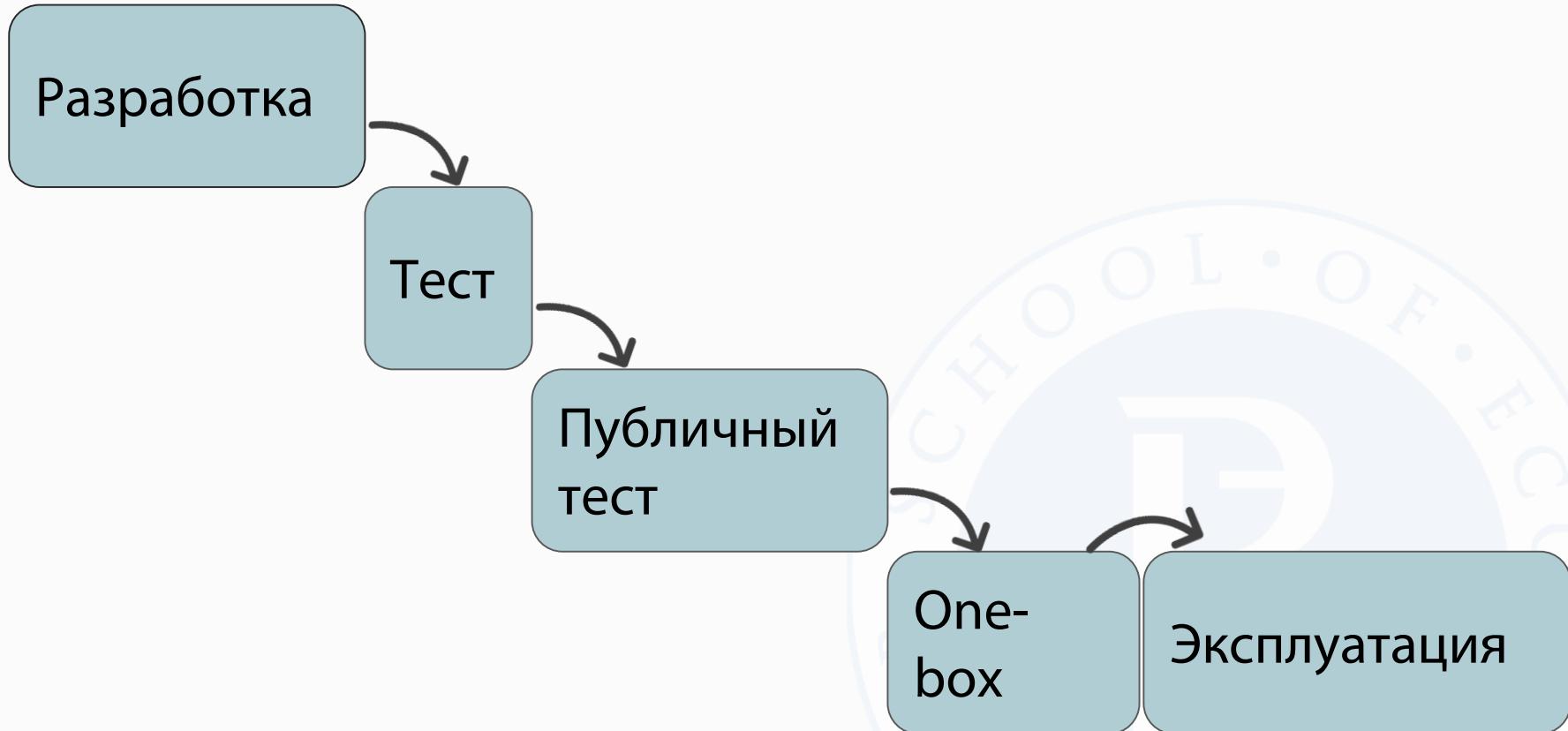
→ Полный цикл внедрения может включать этапы:



# Этапы внедрения в эксплуатацию



Полный цикл внедрения может включать этапы:



# Резюме



Поэтапное внедрение изменений и планирование процесса ввода в эксплуатацию **минимизируют риск недоступности сервиса**



Откат изменений на стабильную версию помогает **снизить негативные последствия** в случае сбоя

# Резюме



Поэтапное внедрение изменений и планирование процесса ввода в эксплуатацию **минимизируют риск недоступности сервиса**



Откат изменений на стабильную версию помогает **снизить негативные последствия** в случае сбоя



Далее: стратегии внедрения различных типов обновлений

# **Типы обновлений сервисов и стратегии внедрения обновлений**

# План



Состояние сервиса и типы обновлений



Стратегии внедрения разных типов обновлений



Rolling release. Непрерывная интеграция изменений

# Состояние ML-сервиса

Сервис хранит модель,  
и она **в процессе** работы сервиса **меняется**,  
сервис хранит состояние — **stateful**



# Состояние ML-сервиса

Сервис хранит модель,  
и она **в процессе** работы сервиса **меняется**,  
сервис хранит состояние — **stateful**



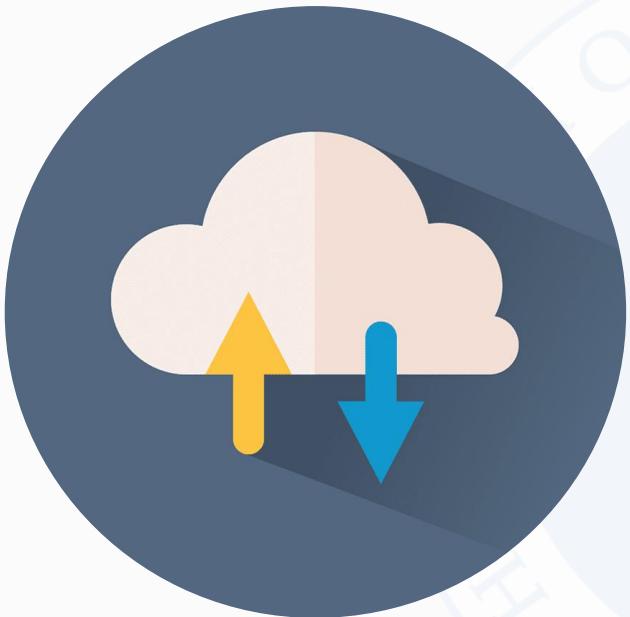
Сервис использует модель, обращаясь  
по API к другой машине,  
сервис не хранит состояние — **stateless**



# Stateful и Stateless обновления

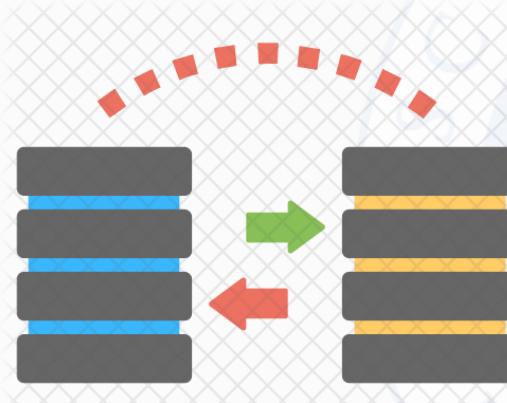


Обновления без состояния — только **код**, происходят быстро



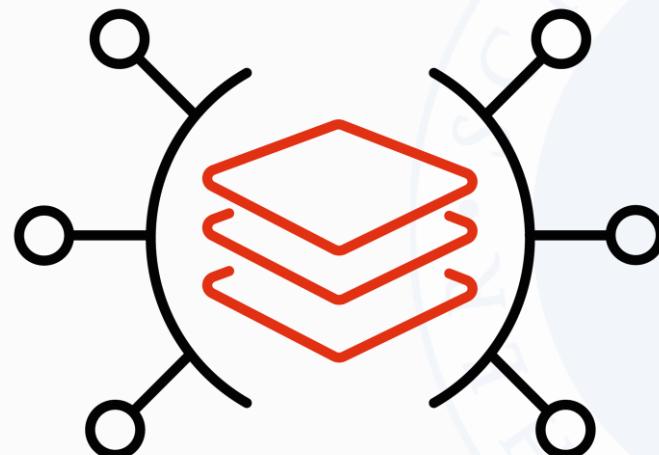
# Stateful и Stateless обновления

- Обновления без состояния — только **код**, происходят быстро
- Обновления с **изменением состояния** — меняются **данные**, необходимо приводить систему в некоторое состояние, могут занимать время



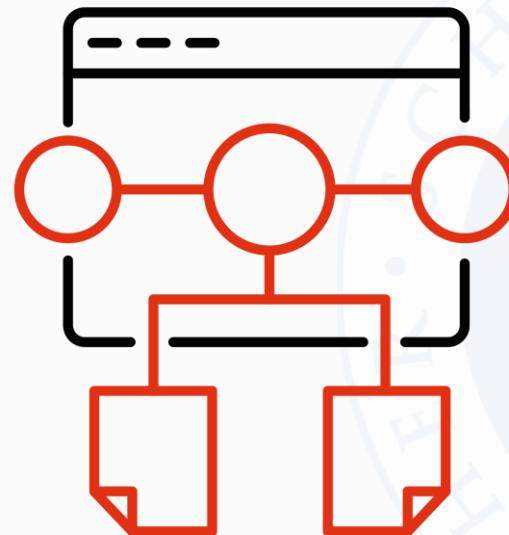
# Внедрение обновлений без состояния

- Достаточно обновить **конфигурацию**, указав новую версию кода/модели
- Если перезапуск занимает **значительное время** по сравнению с частотой запросов, следует использовать **Blue-Green Deployment**



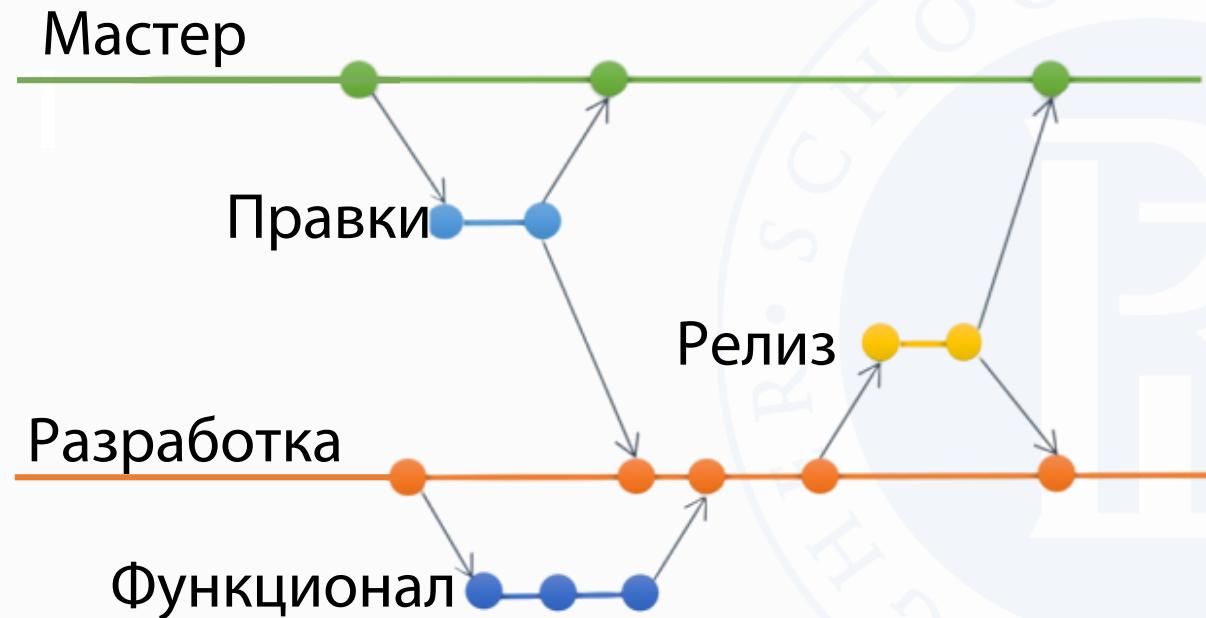
# Внедрение обновлений с состоянием

- Всегда следует использовать Blue-Green Deployment
- Следует проводить дополнительную проверку того, что полученное состояние соответствует ожидаемому



# Подготовка выпуска с рядом изменений

- Процесс обновления может содержать много **ручных действий** и требовать работы инженера
- Изменения **накапливаются**, затем **вводятся разом**



# Подготовка выпуска с рядом изменений

- Процесс обновления может содержать много **ручных действий** и требовать работы инженера
- Изменения **накапливаются**, затем **вводятся разом**

## Плюсы:

- Экономит время инженера по вводу в эксплуатацию

## Минусы:

- Трудно отследить источники ошибок
- Откат всех последних изменений сразу

# Rolling release

- Стратегия внедрения обновлений, небольшие изменения сразу же проходят через процесс интеграции и вводятся в эксплуатацию

# Rolling release

- Стратегия внедрения обновлений, небольшие изменения сразу же проходят через процесс интеграции и вводятся в эксплуатацию
- Требует надежный и автоматизированный процесс ввода изменений в эксплуатацию

# Rolling release

- Стратегия внедрения обновлений, небольшие изменения сразу же проходят через процесс интеграции и вводятся в эксплуатацию
- Требует надежный и автоматизированный процесс ввода изменений в эксплуатацию
- Гранулярность изменений позволяет легко обнаружить причину сбоя

# Непрерывная интеграция

- Rolling release часто используется при [автоматической выкладке](#) изменений

# Непрерывная интеграция

- Rolling release часто используется при [автоматической выкладке](#) изменений
- Регулярный сбор обновлений в централизованном хранилище и [тестирование](#) системы с этими изменениями

# Непрерывная интеграция

- Rolling release часто используется при **автоматической выкладке** изменений
- Регулярный сбор обновлений в централизованном хранилище и **тестирование** системы с этими изменениями
- Каждое изменение запускает процесс тестирования

# Непрерывная интеграция

- Rolling release часто используется при **автоматической выкладке** изменений
- Регулярный сбор обновлений в централизованном хранилище и **тестирование** системы с этими изменениями
- Каждое изменение запускает процесс тестирования
- Непрерывное внедрение — автоматическое введение в эксплуатацию изменений, прошедших процесс тестирования

# Резюме



Состояние сервиса и тип обновления влияют на стратегию обновления



Непрерывная интеграция изменений требует автоматизации процесса, но упрощает работу по введению изменений в эксплуатацию

# Резюме



Состояние сервиса и тип обновления влияют на стратегию обновления



Непрерывная интеграция изменений требует автоматизации процесса, но упрощает работу по введению изменений в эксплуатацию



Далее: планирование ввода несовместимых изменений

# **Обновления с несовместимыми изменениями**



# План



Планирование внедрения несовместимых изменений



Пример обновления структуры БД



Пример обновления API



# Несовместимые изменения

- Изменения компонент сервиса, которые нарушают взаимодействие
- Требуют сложного процесса обновления и отката



# Примеры изменения в структуре БД



Изменение типа поля

Имя	Год
string	uint32

Имя	Год	Дата
string	uint32	timestamp

Имя	Дата
string	timestamp

# Примеры изменения в структуре БД

→ Изменение **типа поля**

→ Изменение **формулы вычисления значения**

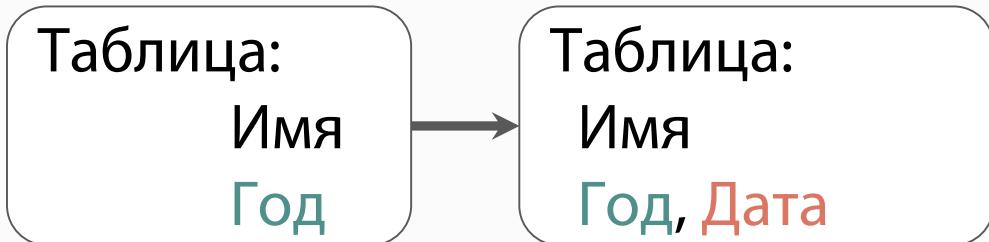
Имя	Год
string	uint32

Имя	Год	Дата
string	uint32	timestamp

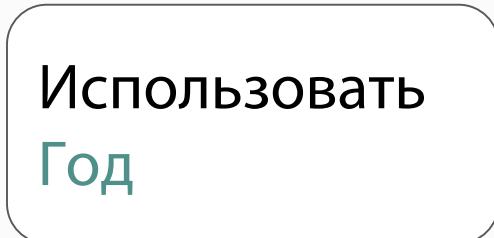
Имя	Дата
string	timestamp

# Обновление БД

База данных

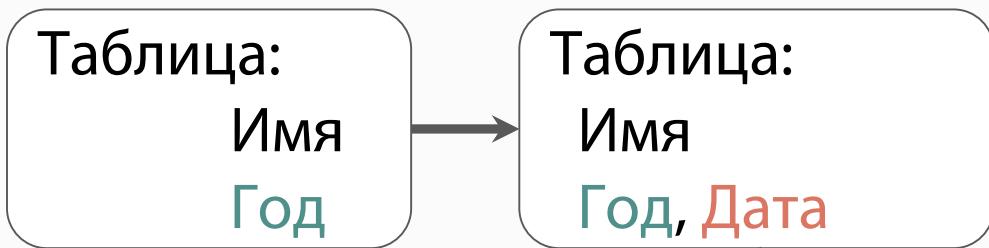


Код

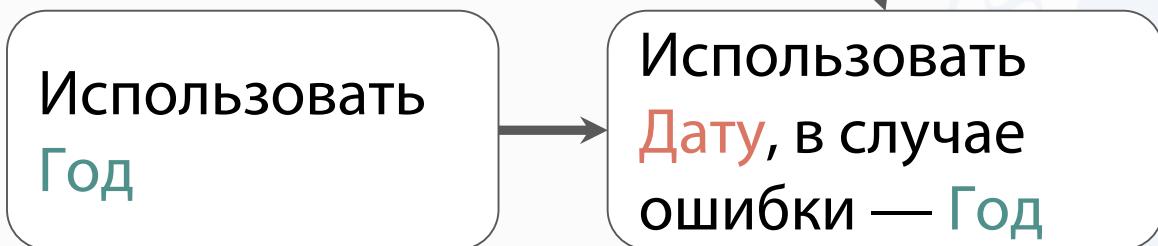


# Обновление БД

База данных

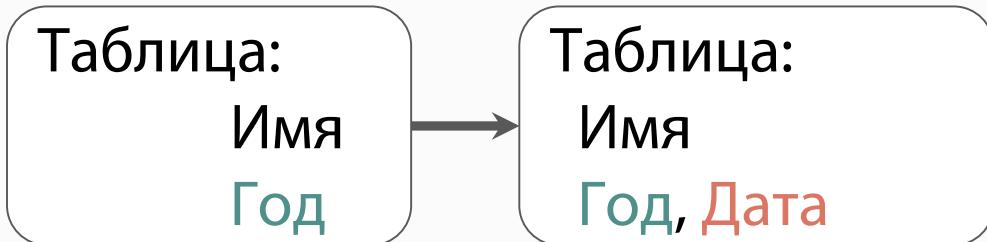


Код

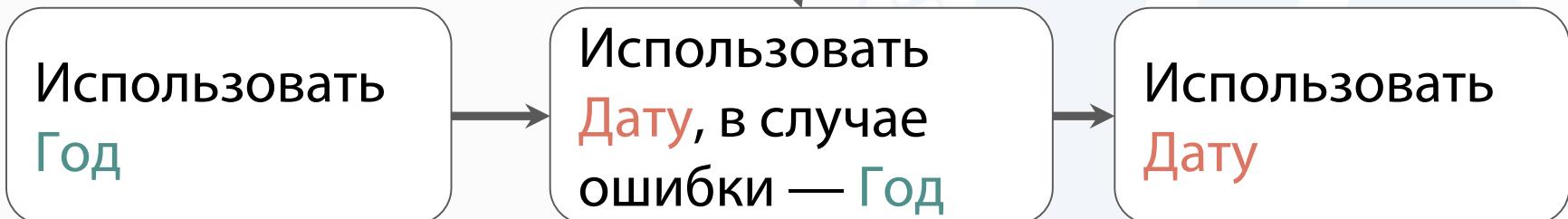


# Обновление БД

База данных

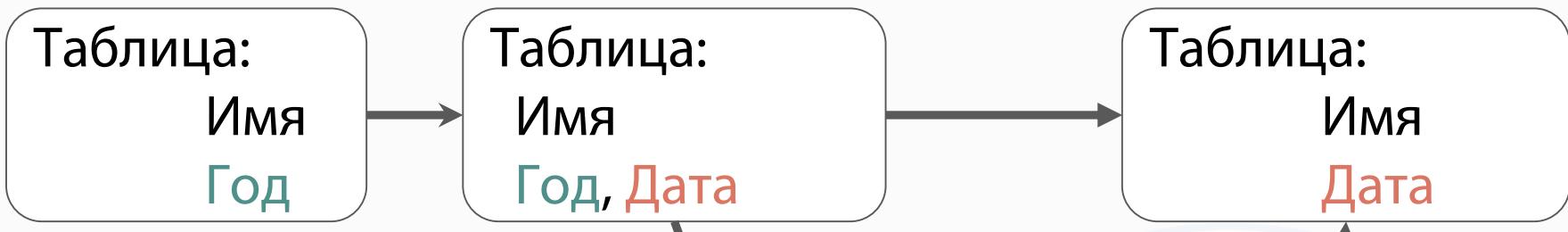


Код

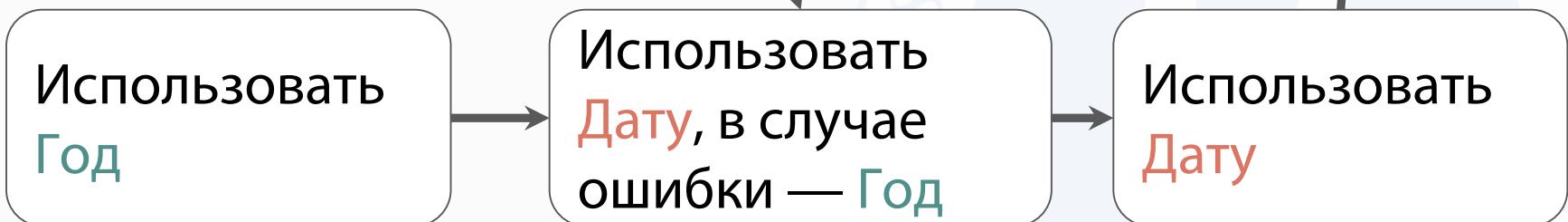


# Обновление БД

База данных

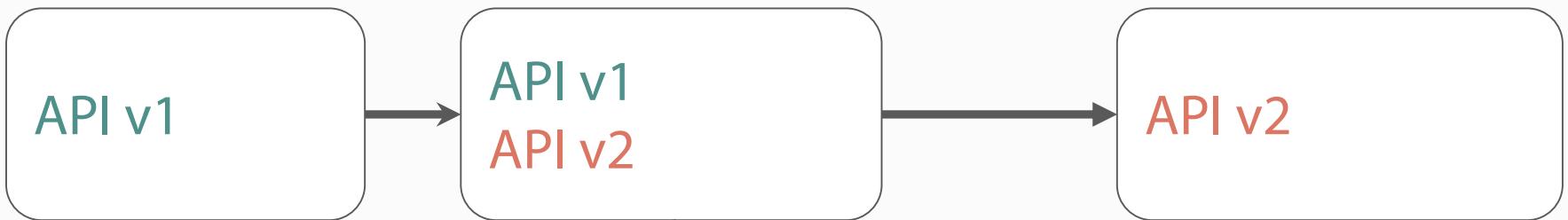


Код

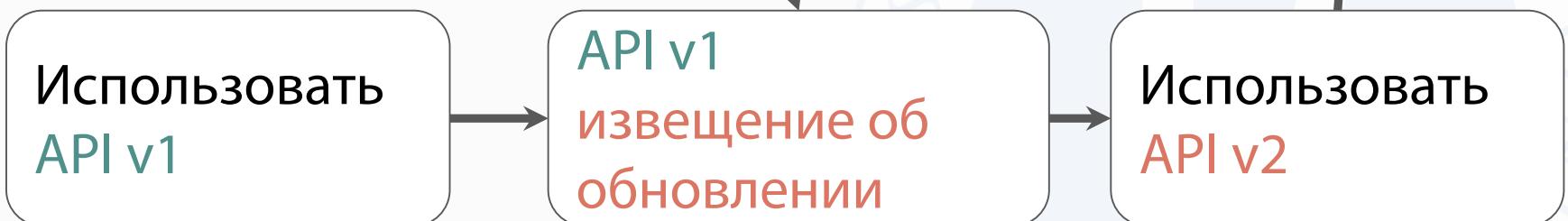


# Обновление API и ввода/вывода

API



Код



# Планирование выкладки

→ Выкладываются **вручную**, либо разбиваются на **простые** изменения и выкладываются **автоматически**



# Планирование выкладки

- Выкладываются **вручную**, либо разбиваются на **простые** изменения и выкладываются **автоматически**
- Нужно планировать запуск скриптов **миграции** и **проверки**, что они сработали



# Планирование выкладки

- Выкладываются **вручную**, либо разбиваются на **простые изменения** и выкладываются **автоматически**
- Нужно планировать запуск скриптов **миграции** и **проверки**, что они сработали
- Нужен **план отката** к предыдущей версии из-за потенциального риска некорректной работы

