

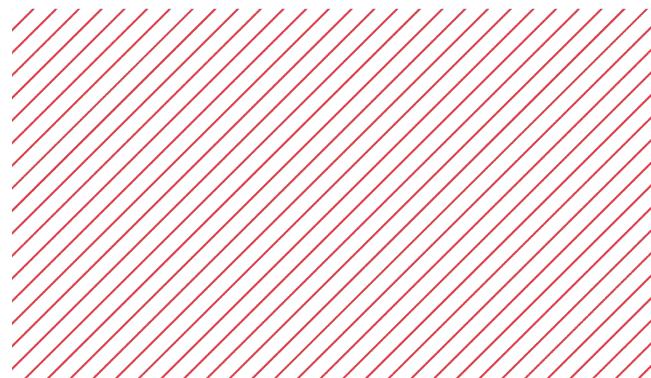
академия
больших
данных



mail.ru
group

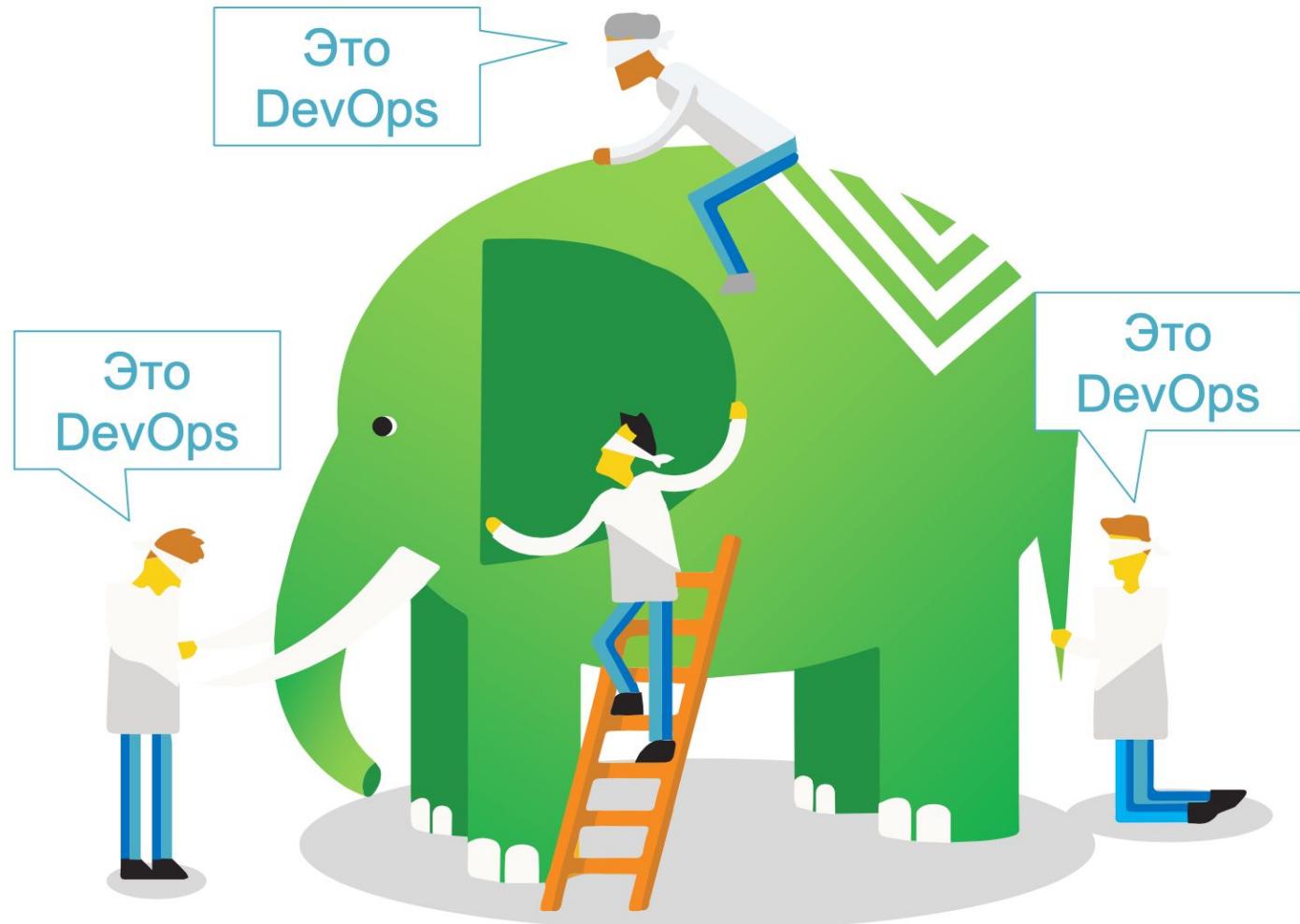
CI/CD для ML моделей

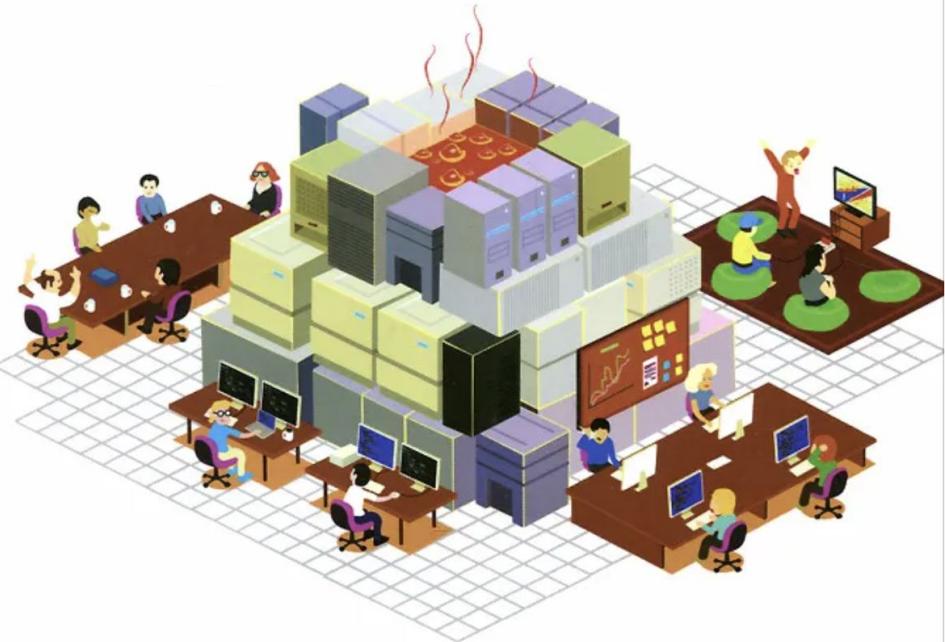
Михаил Марюфич, MLE



Что такое DEVOPS?

- Это устранение барьеров между DEV и OPS
- Автоматизация
- Когда мы часто проверяем код



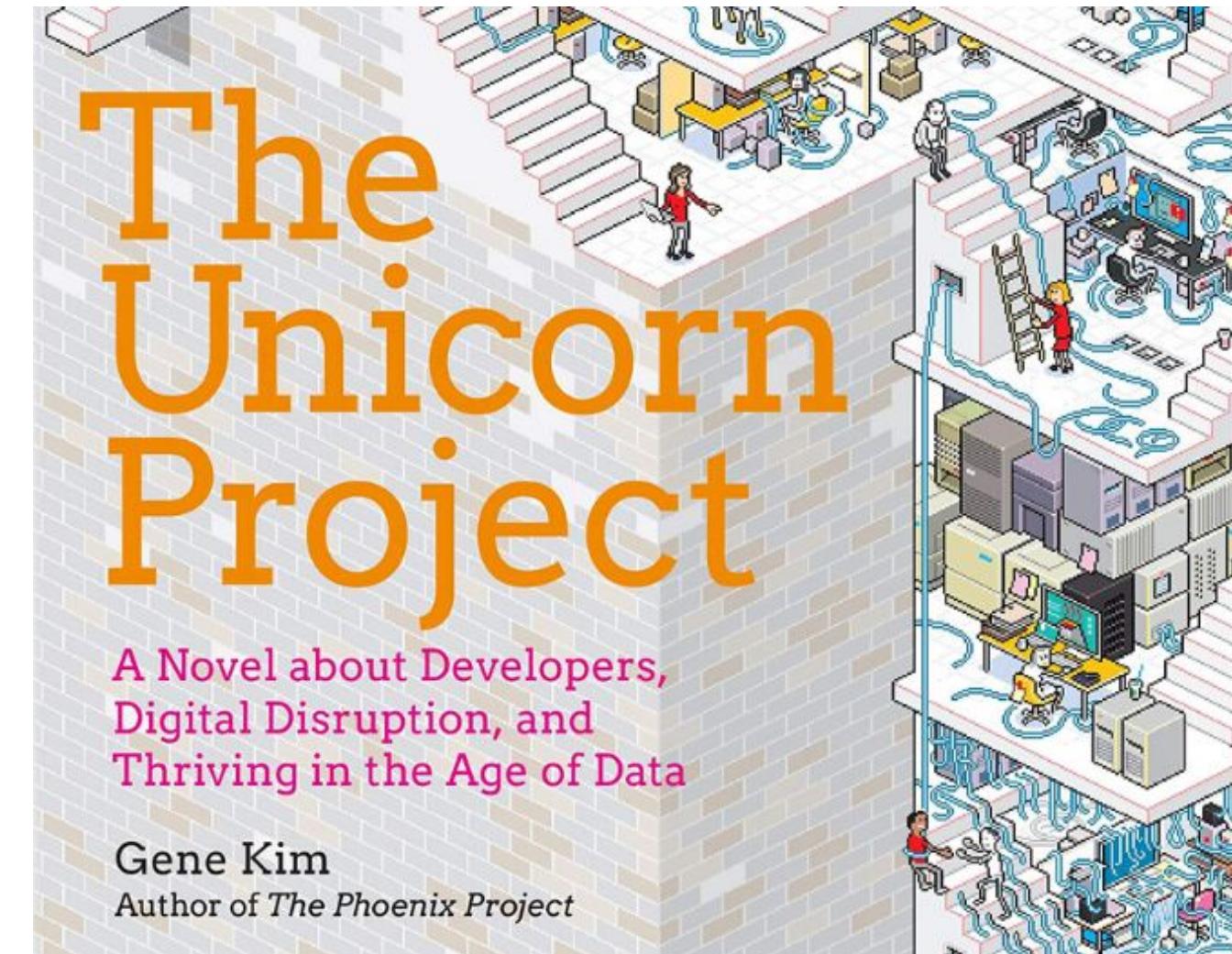


Проект «Феникс»

Роман о том, как **DevOps**
меняет бизнес к лучшему

DevOps – новый метод разработки программного обеспечения, нацеленный на эффективное взаимодействие специалистов. Его цель – помочь компаниям создавать качественные программные продукты и сервисы.

Джин Ким, Кевин Бер, Джордж Спаффорд

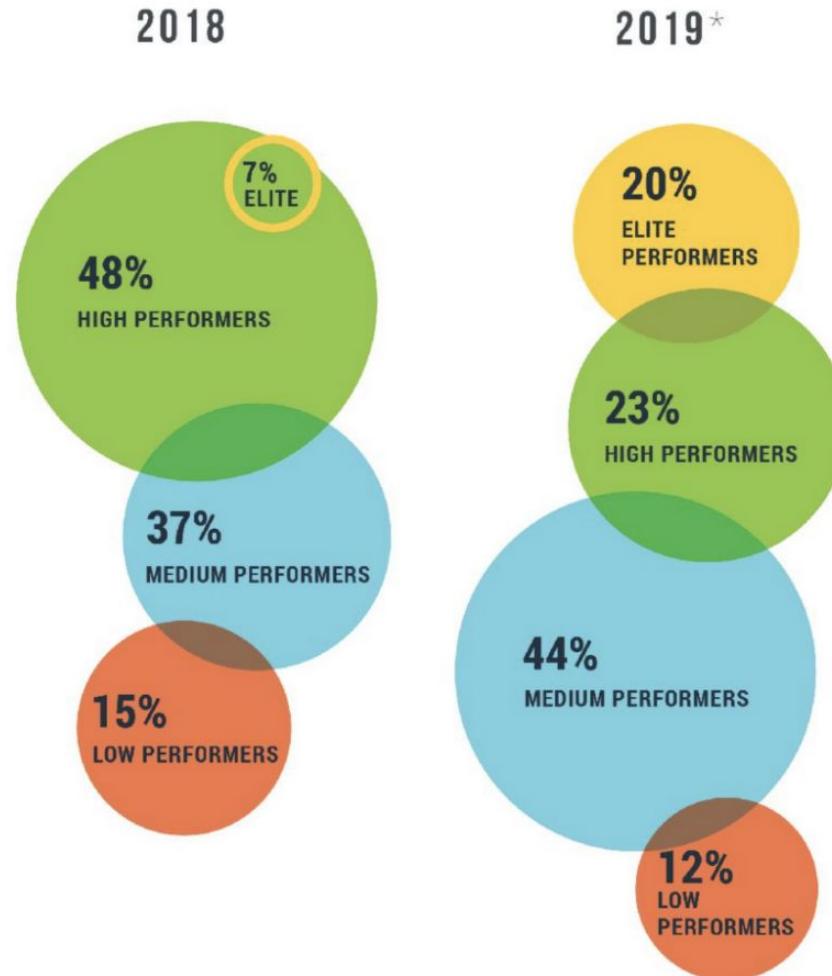




Почему все хотят DevOps?

- Быстрый выход в продакшн (reduce time to market)
- Уменьшение количества сбоев, откатов и времени на восстановление
- Постоянная обратная связь
- Улучшение качества продукта
- Автоматизация монотонных задач
- экономия денег

The 2019 Accelerate State of DevOps



Как круто быть elite performers!

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have...



208
TIMES MORE
frequent code deployments

106
TIMES FASTER
lead time from commit to deploy

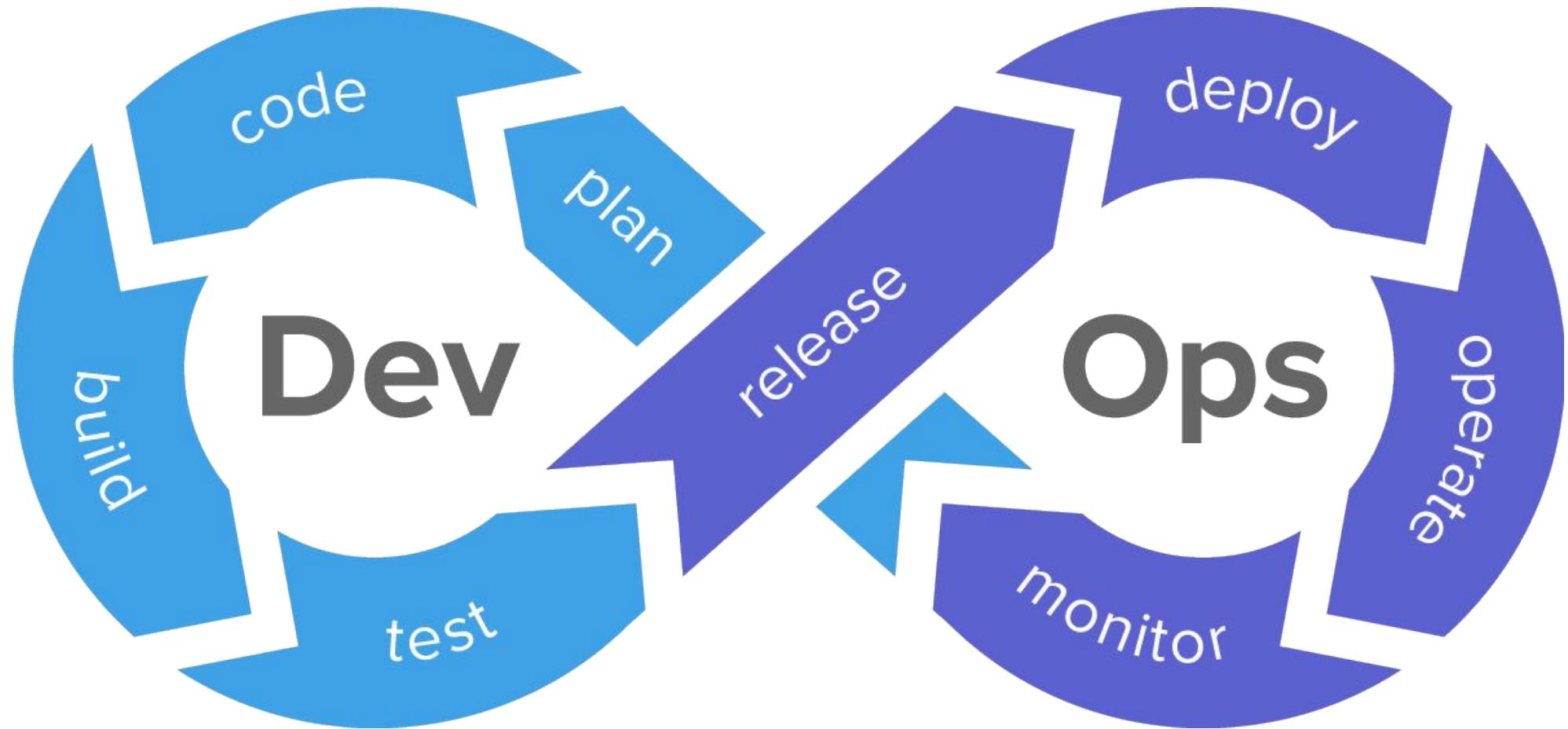


2,604
TIMES FASTER
time to recover from incidents

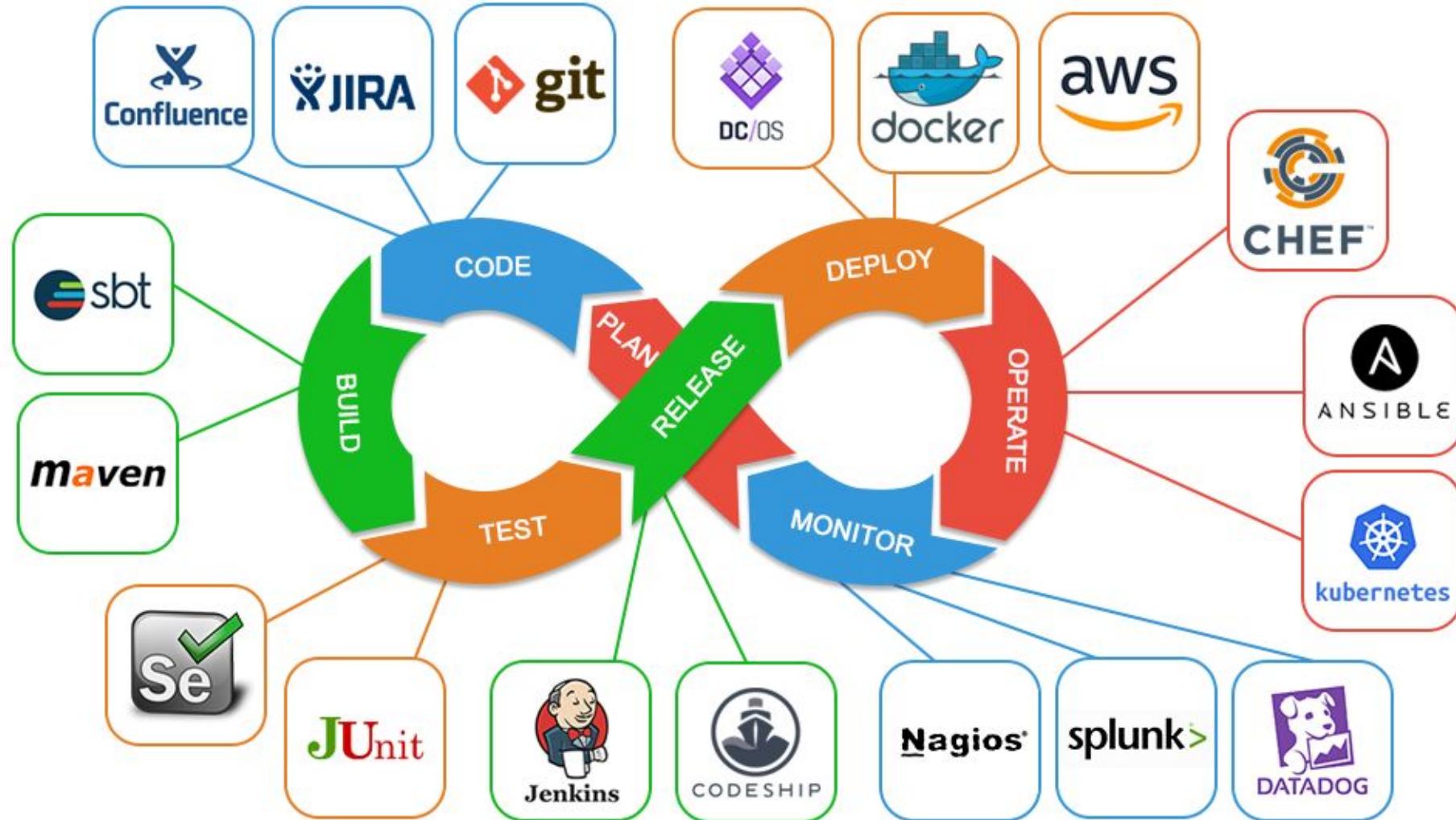
7
TIMES LOWER
change failure rate
(changes are $\frac{1}{7}$ as likely to fail)

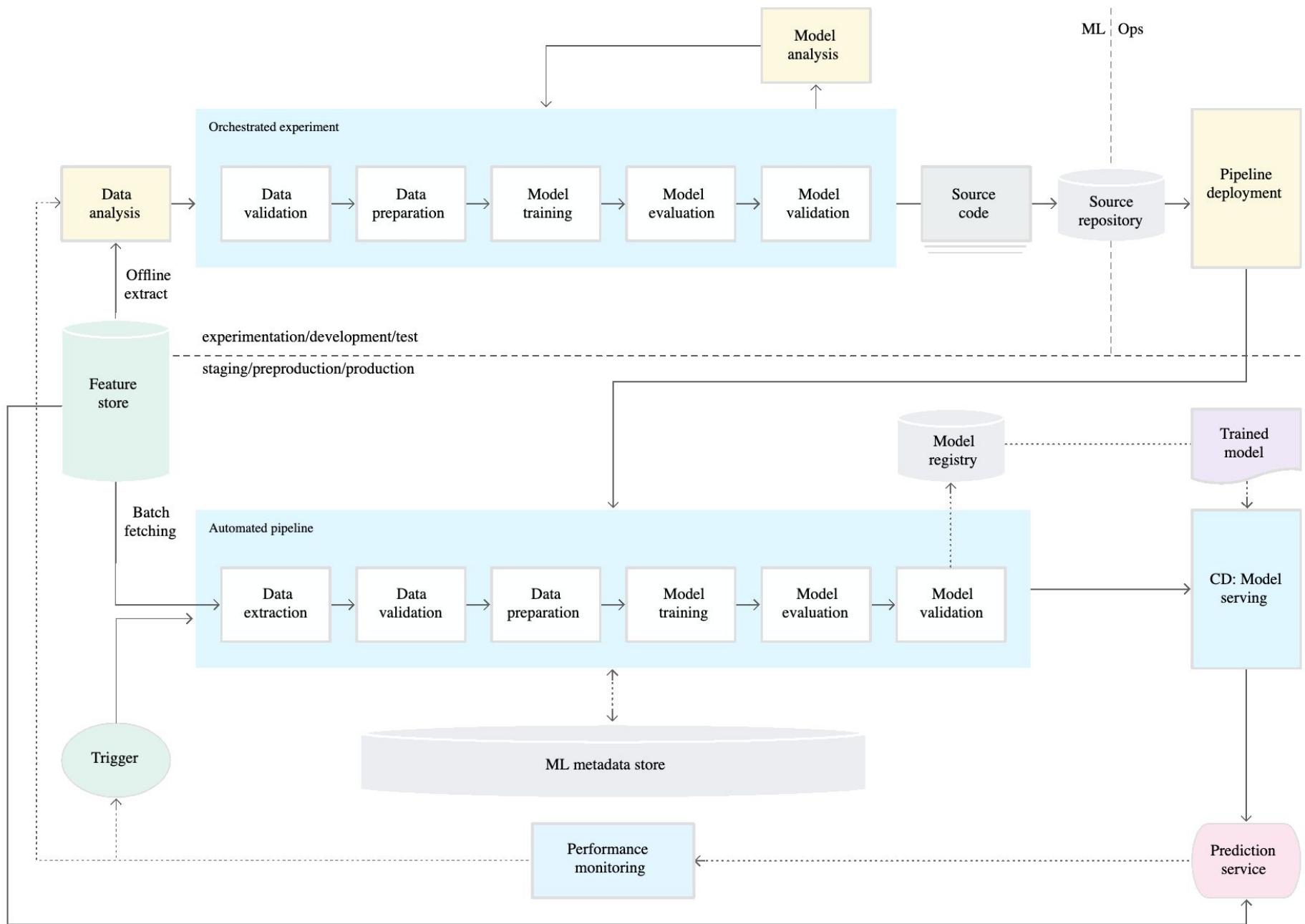


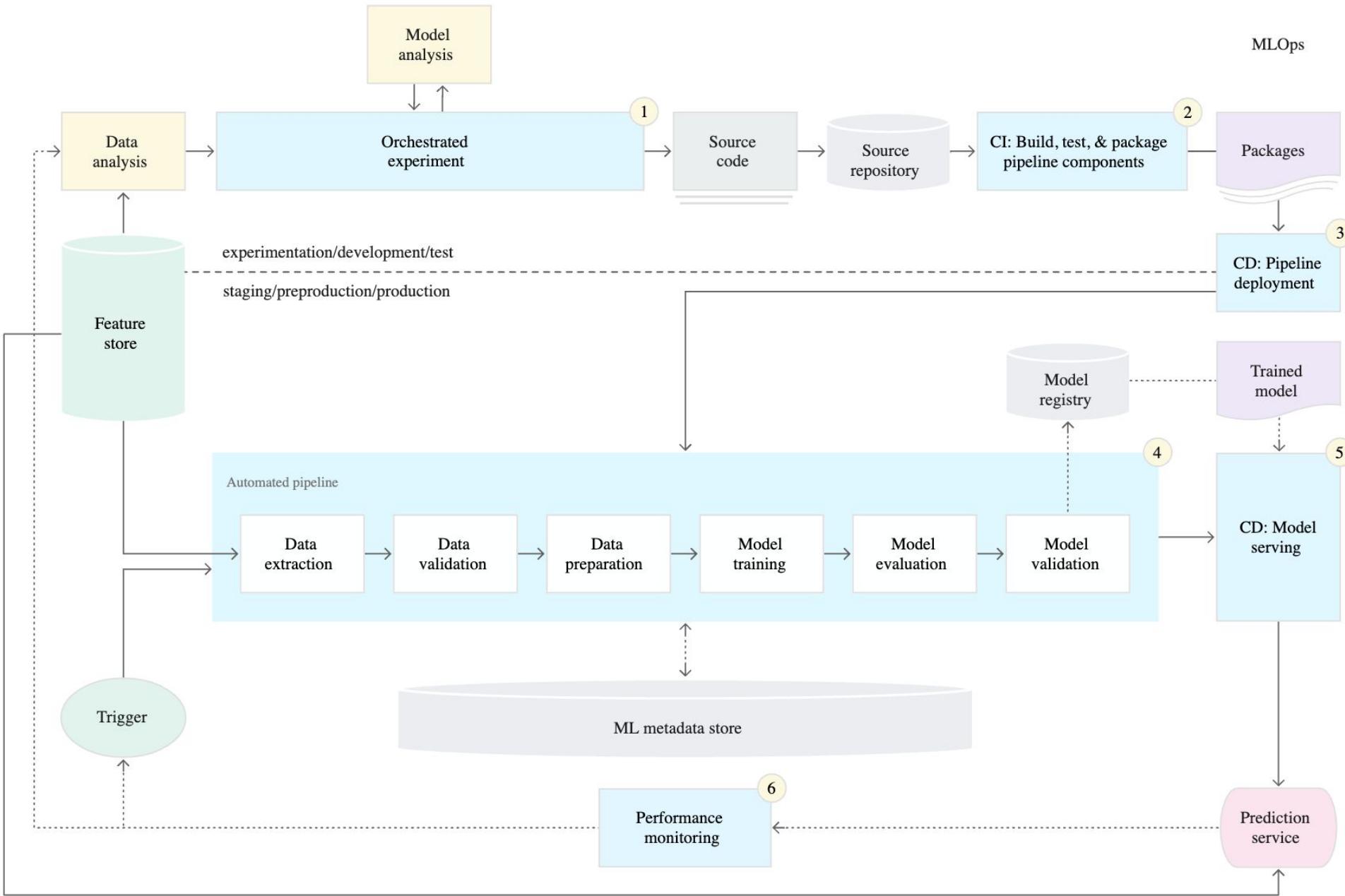
Что такое DEVOPS?



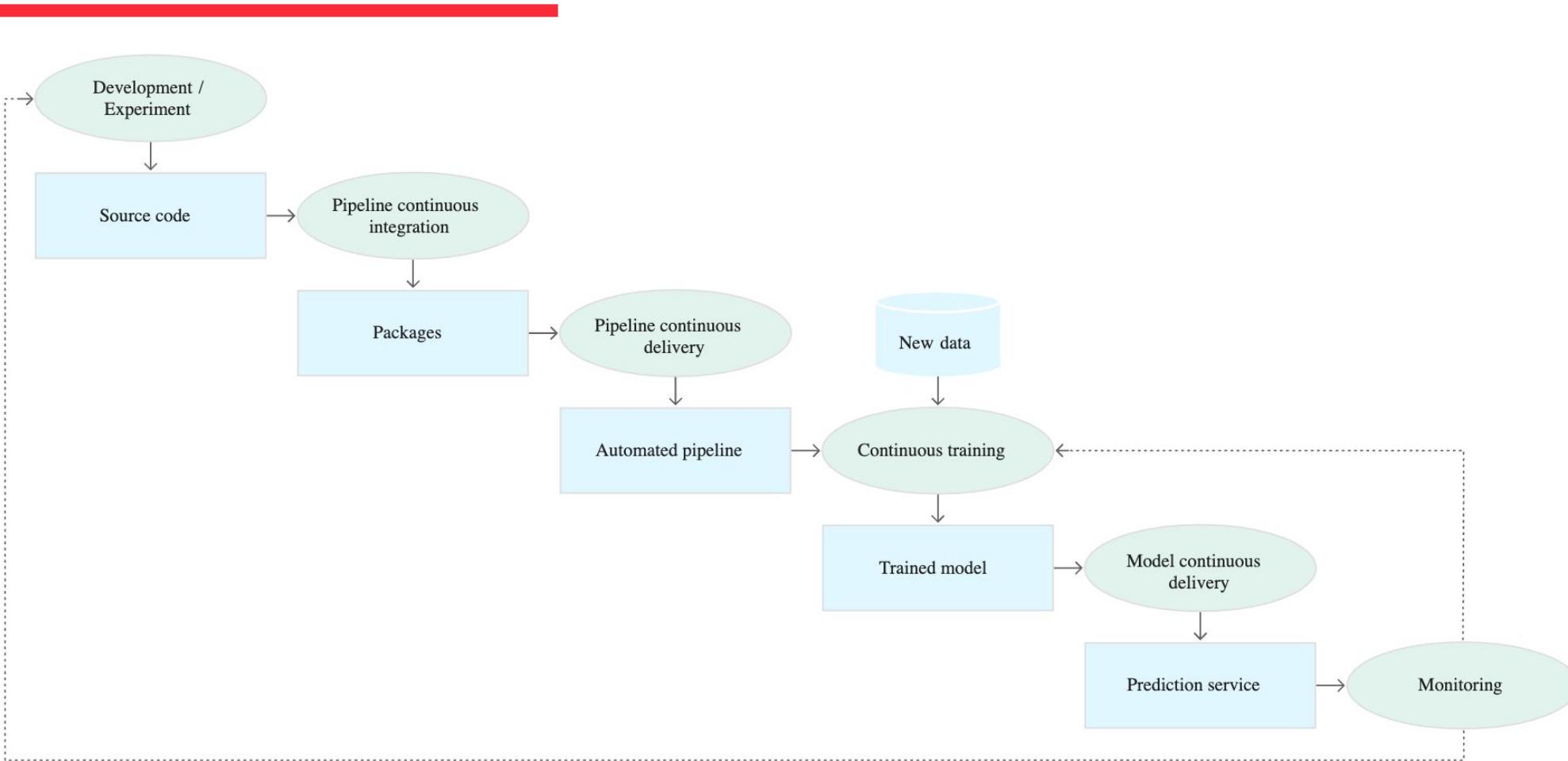
Что такое DEVOPS?







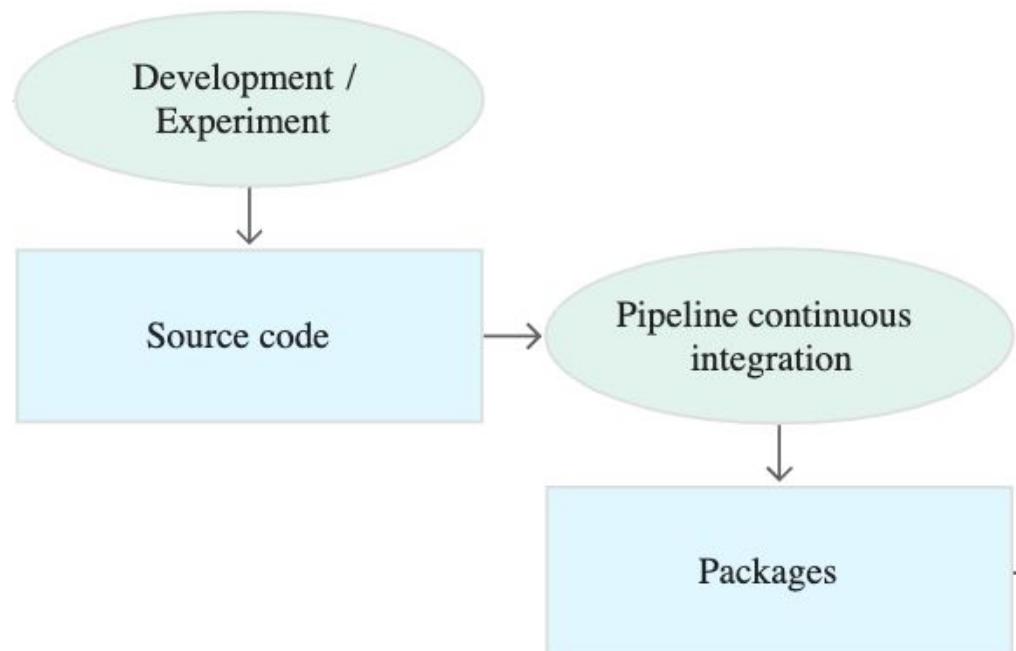
План на сегодня!



Continuous Integrations

Это практика разработки, в которой все участники команды часто интегрируют свою работу в общую кодовую базу.

Каждая интеграция проверяется на наличие ошибок.





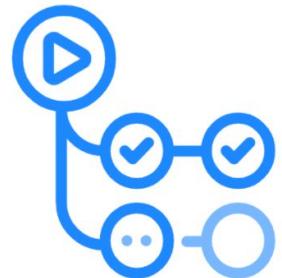
Критерии наличия CI

Все ежедневно мержат свой код в мастер

На каждый коммит запускаются юнит-тесты

Быстрая починка сломанного билда, например, за 10 минут (заметили и починили!)

Инструменты CI/CD



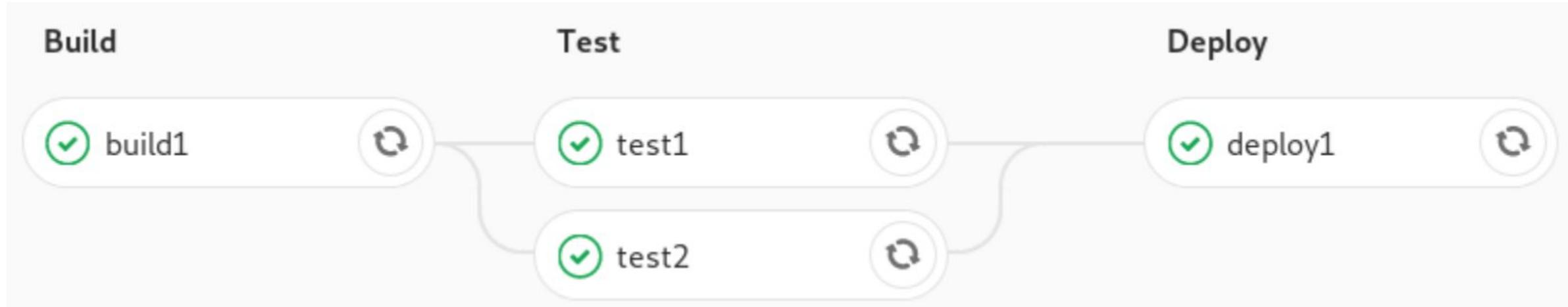
GitHub Actions



GitLab

Функции CI/CD инструмента

- Взаимодействие с VCS
- Позволяет описывать инструкции для пайплайнов сборки, валидации качества тестирования, выкатки приложений





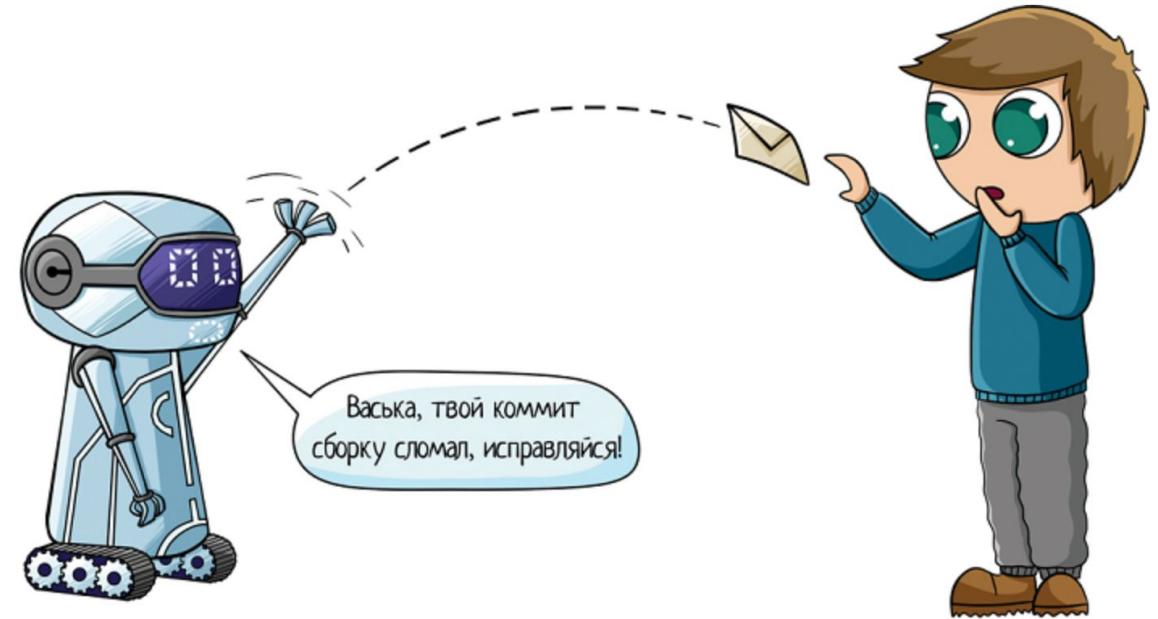
Отличия друг от друга

- платформа
- распространенность
- поддержка
- синтаксис/формат пайплайнов

Типичные задачи CI

Ответить на вопросы:

- А собирается ли код?
- Проходят ли автотесты?
- А достаточно ли он качественный?



GITHUB ACTIONS



GitHub Actions

Automate your workflow from idea to production

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

[Get started with Actions](#)

Questions? [Contact Sales →](#)



<https://github.com/features/actions>

GITHUB ACTIONS — слайд про галочки в PR

	Visibility ▾	Organization ▾	Sort ▾
16 Open ✓ 6 Closed			
made-ml-in-prod-2021/kbrodt Homework3 ✓ hw3 #4 opened 7 hours ago by kbrodt			3
made-ml-in-prod-2021/sergey-msu Homework3 ✓ hw3 #5 opened 2 days ago by sergey-msu			9
made-ml-in-prod-2021/reconrus Homework2 hw2 #3 opened 6 days ago by reconrus			
made-ml-in-prod-2021/robertspb Homework2 hw2 #3 opened 6 days ago by robertspb			1
made-ml-in-prod-2021/zingykizz Homework2 hw2 #3 opened 6 days ago by ZingyKizz			1

<https://github.com/features/actions>

```
background-color: #fff;
text-shadow: 0px -1px 0px #000;
filter: dropshadow(color:#000);
color:#777;

}

header #main-navigation ul li span:hover,
header #main-navigation ul li span:active {
  border: 1px solid #000;
  background-color: #F9F9F9;
  box-shadow: 0px 0px 1px #000;
  -webkit-box-shadow: 0px 0px 2px #000;
  -moz-box-shadow: 0px 0px 1px #000;
}
```

DEMO прогоним линтер/тесты
на python проекте



Термины

Workflow — набор инструкций

Job — набор step в рамках workflow, запускаемых на одном runner

Step — job состоит из них

Event — действие(создали пулл реквест, сделали коммит, etc)

Action — переиспользуемый unit

Типы событий

На создание/закрытие пулл реквестов,
пуши, добавление ревьюеров и многое
другое

Scheduled events

schedule

Manual events

workflow_dispatch

repository_dispatch

Webhook events

check_run

check_suite

create

delete

deployment

deployment_status

fork

gollum

issue_comment

```
background-color: #F5F5F5;
text-shadow: 0px -1px 0px #F5F5F5;
filter: dropshadow(color:#777;
color:#777;

}
header #main-navigation ul li span:hover,
box-shadow: 0px 0px 1px #F5F5F5;
-webkit-box-shadow: 0px 0px 2px #F5F5F5;
moz-box-shadow: 0px 0px 1px #F5F5F5;
background-color:#F9F9F9;
active span,
li span.dashboard,
...[img/dashboard.png] span,
li span.dashboard,
...[img/dashboard.png] span
```

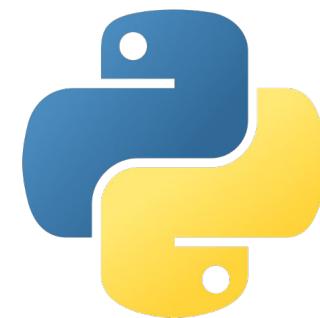
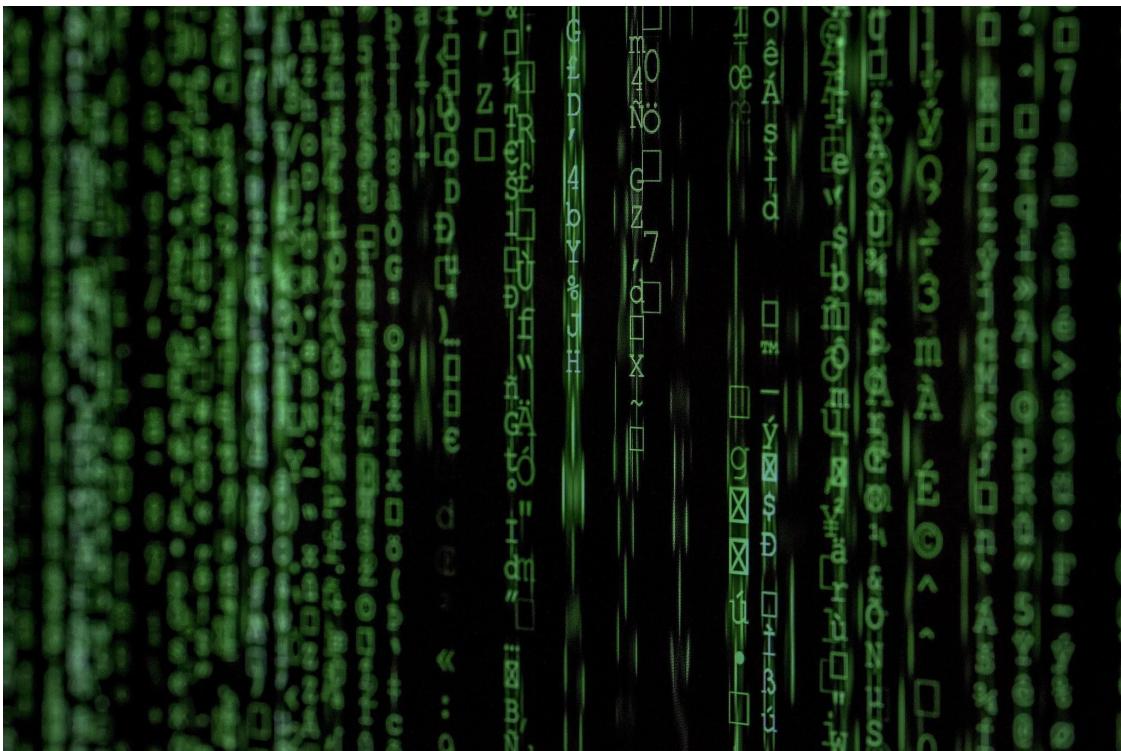
Смотрим workflow повнимательнее

Action

Переиспользуемый компонент, можно писать свои такие

```
- uses: actions/checkout@v2
- name: Set up Python ${{ matrix.python-version }}
  uses: actions/setup-python@v2
  with:
    python-version: ${{ matrix.python-version }}
```

Жизненный цикл python package



рурі

Workflow для публикации

```
name: Upload Python Package

on:
  release:
    types: [created]

jobs:
  deploy:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.x'
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install setuptools wheel twine
      - name: Build and publish
        env:
          TWINE_USERNAME: ${{ secrets.PYPI_USERNAME }}
          TWINE_PASSWORD: ${{ secrets.PYPI_PASSWORD }}
        run: |
          python setup.py sdist bdist_wheel
          twine upload dist/*
```

Секреты

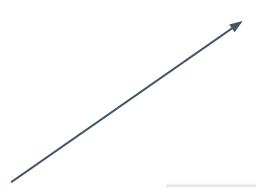
Логины, пароли, явки — всё то, что мы не можем напрямую хранить в системе контроля версий

```
name: Upload Python Package

on:
  release:
    types: [created]

jobs:
  deploy:
    runs-on: ubuntu-latest

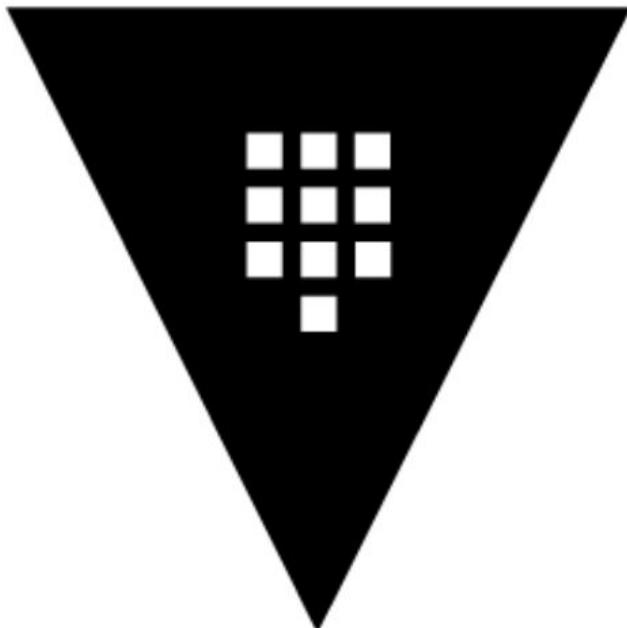
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.x'
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install setuptools wheel twine
      - name: Build and publish
        env:
          TWINE_USERNAME: ${{ secrets.PYPI_USERNAME }}
          TWINE_PASSWORD: ${{ secrets.PYPI_PASSWORD }}
        run: |
          python setup.py sdist bdist_wheel
          twine upload dist/*
```





Где хранить секреты?

Самое известное и популярное
хранилище секретов



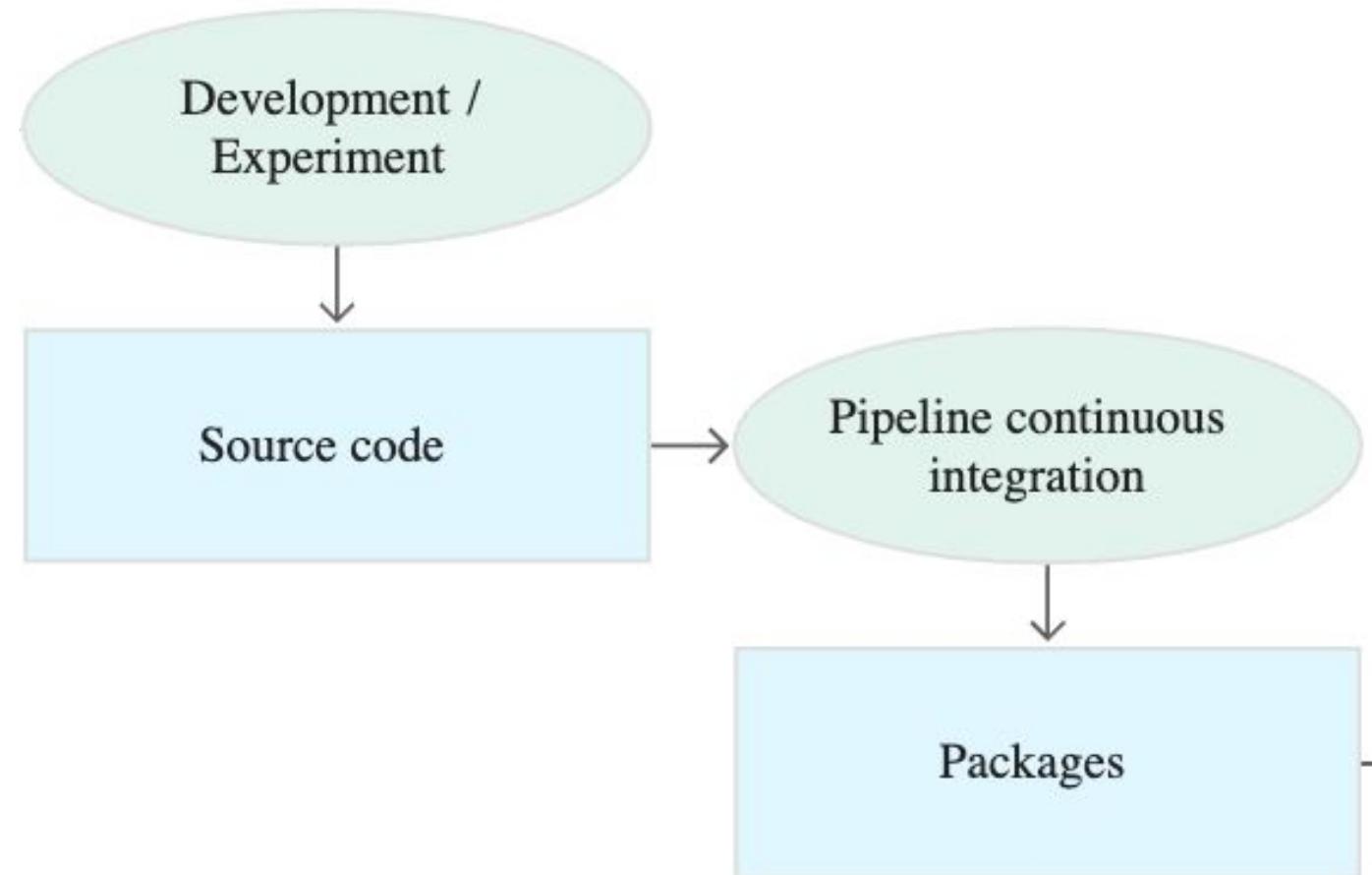
HashiCorp
Vault

DEMO

Создаем секрет и публикуем
пакет в PYPI

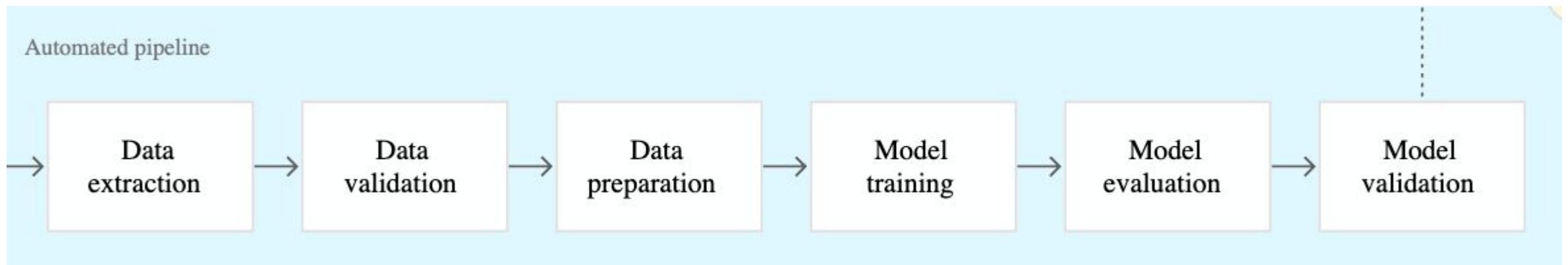
Итог CI пакетов

- 1) Прогоняем тесты, линтеры, ревьюим код
- 2) Публикуем пакеты в registry



Machine Learning Pipeline

Каждый кубик — это переиспользуемый компонент/отдельная JOB/отдельный Docker Image



Оформляем job

Устанавливаем наши пакеты, далее детали реализации
Каждый job что-то считывает с s3 и что-то пишет в s3

9 lines (7 sloc) | 222 Bytes

```
1 FROM python:3.6-slim-stretch
2 RUN mkdir /job/
3 COPY requirements.txt /job/requirements.txt
4 RUN pip install --no-cache-dir -r /job/requirements.txt
5 COPY train.py /job/train.py
6
7 WORKDIR /job
8
9 ENTRYPOINT ["python", "train.py"]
```

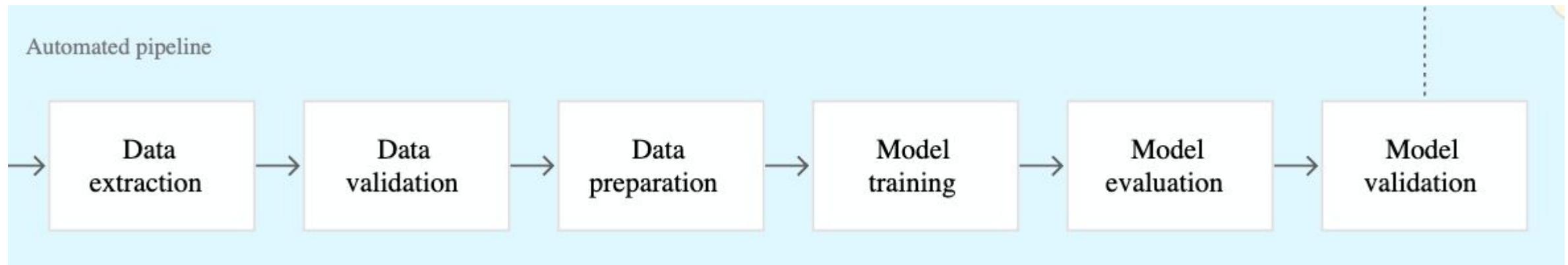
DEMO

Собираем и публикуем докер образ с джобой

<https://github.com/demo-ml-cicd/training-job>

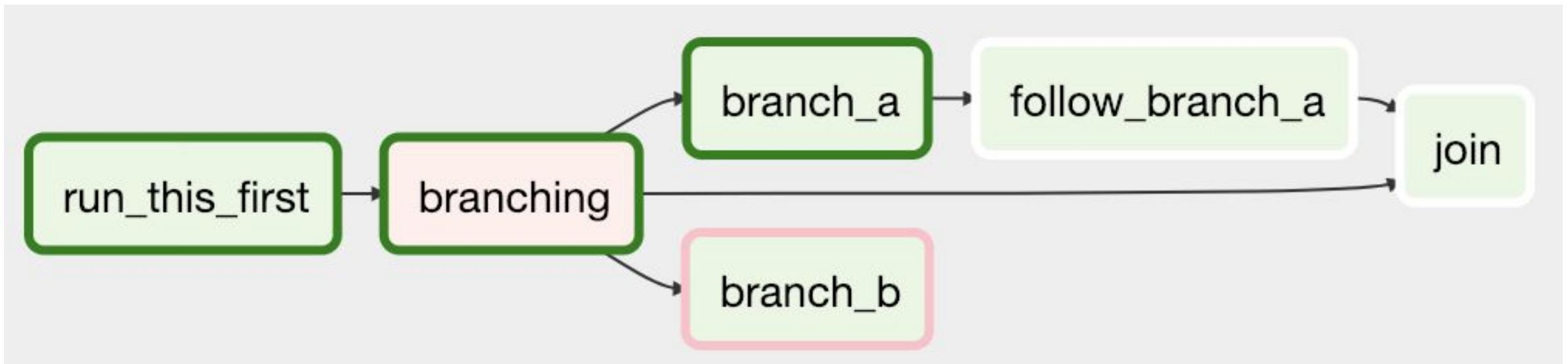
Machine Learning Pipeline

Каждый кубик — это переиспользуемый компонент/отдельная JOB/отдельный Docker Image



Как способ реализовать automated пайплайн мы использовали airflow

AIRFLOW

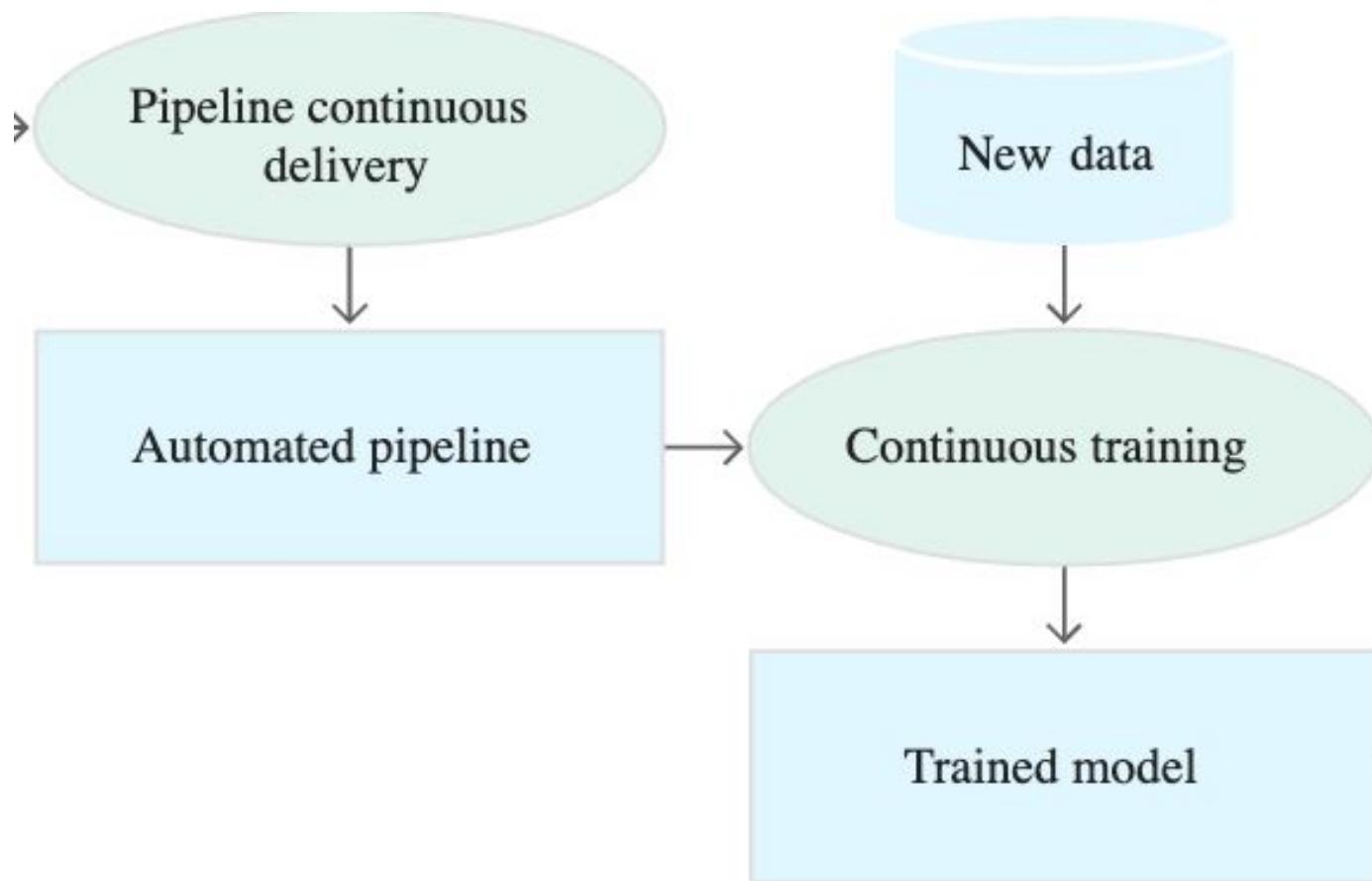


Каждая таска — это **KubernetesPodOperator**

Смотрим на DAG

https://github.com/demo-ml-cicd/airflow-ml-pipelines/blob/stage/dags/train_demo_model_on_k8s.py

Переходим к CD



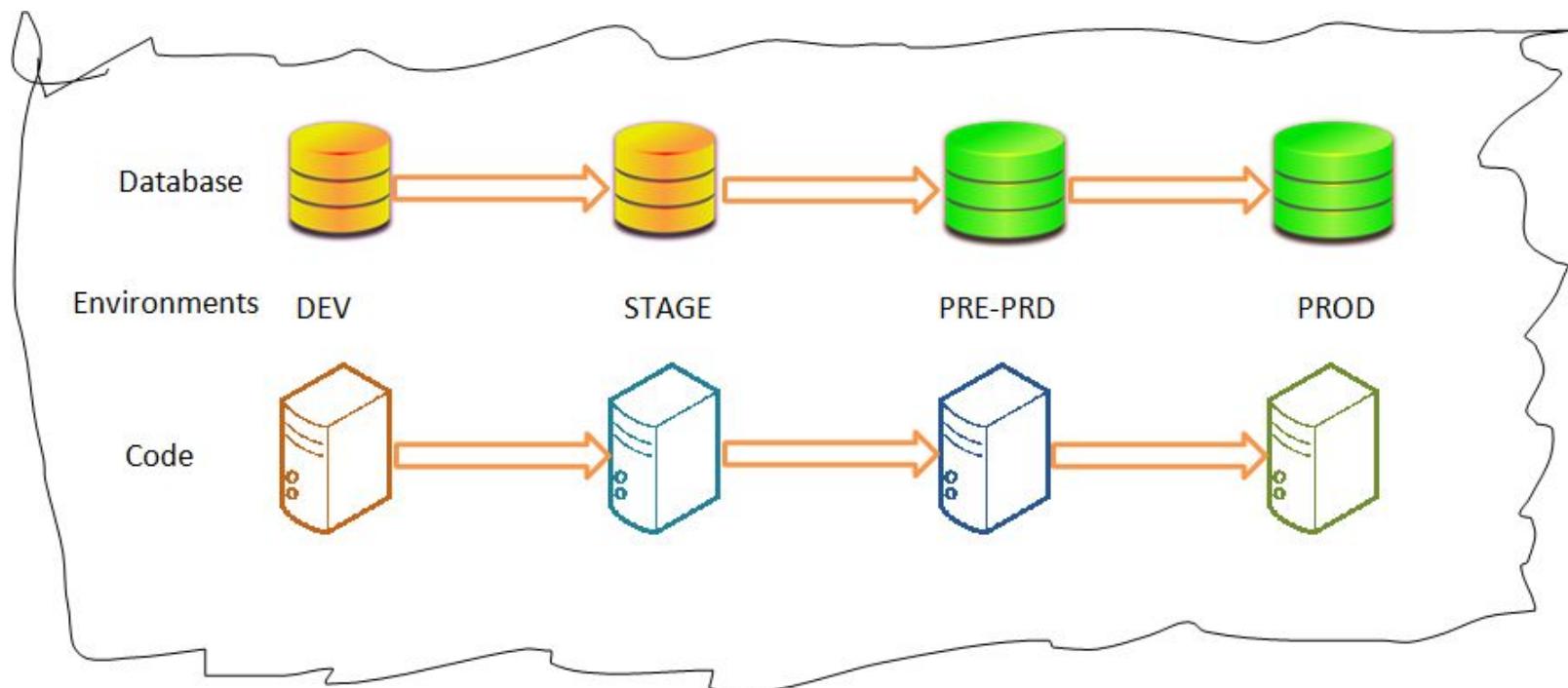


Continuous Delivery

Процесс разработки и эксплуатации ПО, при котором каждое изменение может быть **автоматически** выкачено в боевую среду

Среды

Production environment
Staging environment
Development environment

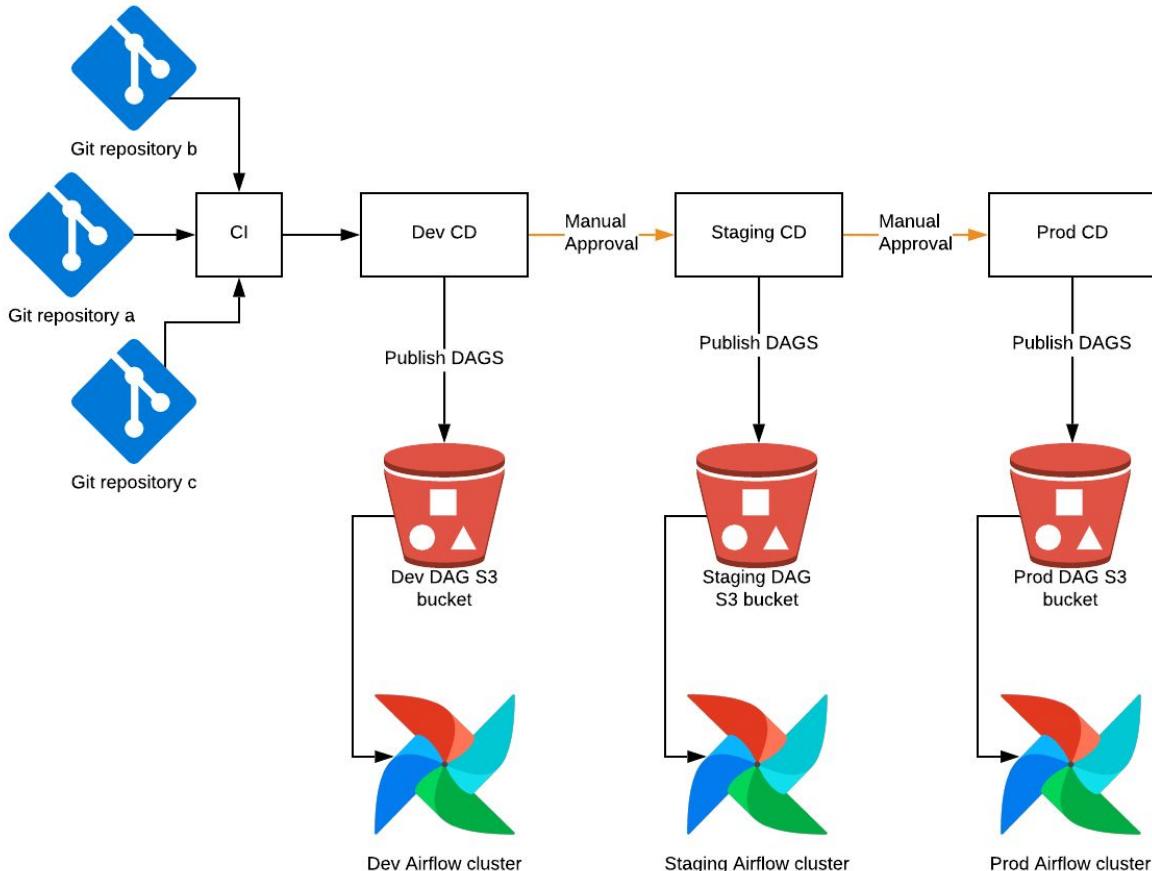


Без Continuous Delivery

надо сходить руками на каждый сервер
удалить старую версию приложения
поставить новую, надеясь, что ничего не забыл

перезапусти его помолясь 10:07

Вариант доставки дагов



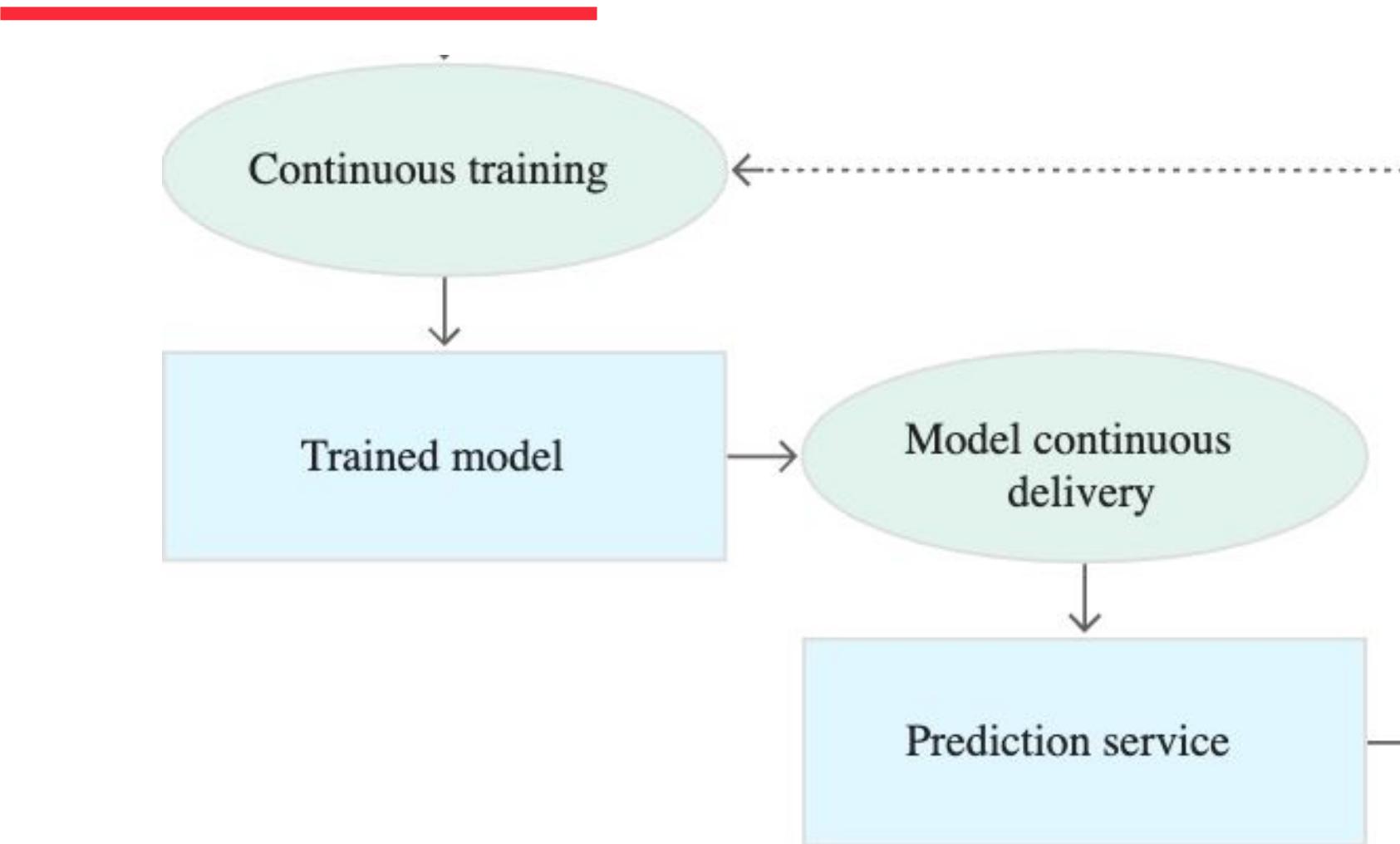
```
background-color: #F5F5F5;
text-shadow: 0px -1px 0px #EAEAEA;
filter: dropshadow(color:#777;
color:#777;

}
header #main-navigation ul li span:hover,
box-shadow: 0px 0px 1px #EAEAEA;
-webkit-box-shadow: 0px 0px 2px #EAEAEA;
moz-box-shadow: 0px 0px 1px #EAEAEA;
background-color:#F9F9F9;
```

DEMO

Выкапываем даги в репозиторий с дагами в airflow, они подтягиваются в облаке
Ну и модельку обучаем

Мы сейчас тут



```
background-color: #fff;
text-shadow: 0px -1px 0px #000;
filter: dropshadow(color:#000);
color:#777;

}

header #main-navigation ul li span:hover,
header #main-navigation ul li span:active {
  border: 1px solid #000;
  background-color: #f9f9f9;
  box-shadow: 0px 0px 1px #000;
  -webkit-box-shadow: 0px 0px 2px #000;
  -moz-box-shadow: 0px 0px 1px #000;
}
```

DEMO: собираем образ с сервером

```
background-color: #F5F5F5;
text-shadow: 0px -1px 0px #EAEAEA;
filter: dropshadow(color:#777;
color:#777;

}
header #main-navigation ul li span:hover,
box-shadow: 0px 0px 1px #EAEAEA;
-webkit-box-shadow: 0px 0px 2px #EAEAEA;
moz-box-shadow: 0px 0px 1px #EAEAEA;
background-color:#F9F9F9;
active span;
```

DEMO:

Выкатываем модельку в кубер



Проблемы

- Нужно атомарно выложить несколько изменения и также их откатить.
- Манифесты, которые мы пишем — похожи друг на друга, их нужно писать много (деплой похожих приложений, деплой в разные среды)
- Иногда хочется установить чужое приложение и не писать никаких манифестов.

HELM

- Шаблонизатор
- Упаковщик



The
package manager
for Kubernetes

Возможности HELM

Собирает несколько манифестов в пакет — **Chart**

Его можно установить, установленный **Chart** — **release**
Во время установки можно подставлять значения

Релиз можно установить (**upgrade**) и откатить (**rollback**)



Ключевые слова

- Chart
- Values
- Release

Release = Chart + Values

Chart

Содержит метаданные, шаблоны описания ресурсов Kubernetes, конфигурация установки (values.yaml),

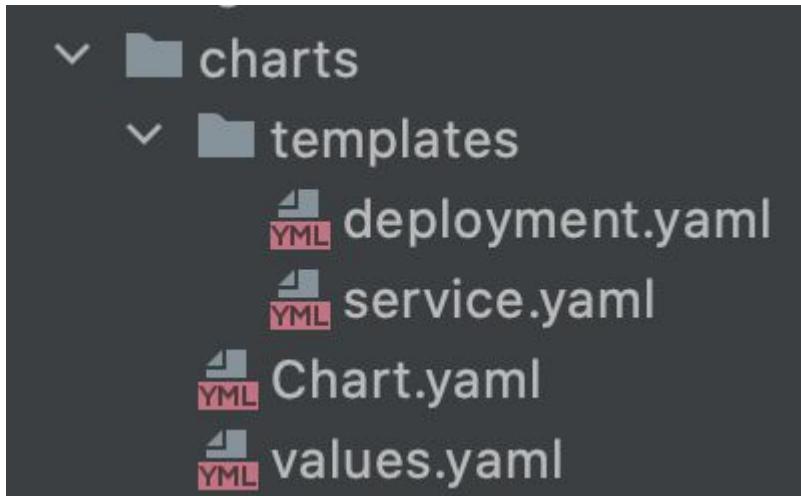


Chart.yaml

Описание пакета

```
apiVersion: v2
name: prediction-service
description: A Helm chart for deploying ml model

type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.1.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
appVersion: 0.0.1
```

Values.yaml

Используются в шаблонах

Могут иметь значения по
умолчанию

```
image:  
  tag: pr-1  
  
replicas: 1  
maxSurge: 2  
maxUnavailable: 2  
  
service:  
  type: LoadBalancer  
  port: 8000  
  targetPort: 8000
```

Helm workflow

```
$ helm install <chart_name> --name=<release_name> --namespace=<namespace> # Создание release
```

```
$ kubectl get secrets -n <namespace> | grep <release_name>
sh.helm.release.v1.<release_name>.v1      helm.sh/release.v1    1    115m
```

```
$ helm upgrade <release_name> <chart_name> --namespace=<namespace> # Обновление release
```

```
$ kubectl get secrets -n <namespace> | grep <release_name>
sh.helm.release.v1.<release_name>.v1      helm.sh/release.v1    1    115m
sh.helm.release.v1.<release_name>.v2      helm.sh/release.v1    1    56m
```

```
$ helm upgrade --install <release_name> <chart_name> --namespace=<namespace> #
Создание или обновление release
```

```
$ kubectl get secrets -n <namespace> | grep <release_name>
sh.helm.release.v1.<release_name>.v1      helm.sh/release.v1    1    115m
sh.helm.release.v1.<release_name>.v2      helm.sh/release.v1    1    56m
sh.helm.release.v1.<release_name>.v3      helm.sh/release.v1    1    5s
```

Templating

Используется шаблонизатор из до

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prediction-service
  labels:
    app: prediction-service
spec:
  replicas: {{ .Values.replicas }}
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: {{ .Values.maxSurge }}
      maxUnavailable: {{ .Values.maxUnavailable }}
```

Templating – страшненький пример

```
{%- end %}

{{- if and .Values.dags.gitSync.enabled .Values.dags.gitSync.knownHosts --}}
  known_hosts: |
    {{ .Values.dags.gitSync.knownHosts | nindent 4 }}
{{- end %}}
{{- if or (eq $.Values.executor "KubernetesExecutor") (eq $.Values.executor "CeleryKubernetesExecutor") --}}
{{- if semverCompare ">=1.10.12" .Values.airflowVersion --}}
  pod_template_file.yaml: |-
{{- if .Values.podTemplate --}}
  {{ .Values.podTemplate | nindent 4 }}
{{- else --}}
  {{ tpl (.Files.Get "files/pod-template-file.kubernetes-helm-yaml") . | nindent 4 }}
{{- end --}}
{{- end %}}
{{- if .Values.kerberos.enabled --}}
  krb5.conf: |
    {{ tpl .Values.kerberos.config . | nindent 4 }}
{{- end --}}
{{- end %}}
```

```
background-color: #F5F5F5;
text-shadow: 0px -1px 0px #EAEAEA;
filter: dropshadow(color:#777);
color:#777;

}

header #main-navigation ul li span:hover,
0px 0px 1px #F5F5F5,
-webkit-box-shadow: 0px 0px 2px #F5F5F5,
moz-box-shadow: 0px 0px 1px #F5F5F5;
background-color:#F9F9F9;
```

DEMO:

Выкатываем модельку с
помощью HELM

Нужно настроить CD пайплайн

Manually triggered 1 minute ago Status Total duration Artifacts

Mikhail-M ➔ addc885 Waiting - -

💡 Mikhail-M requested your review to deploy to PRODUCTION Review deployments

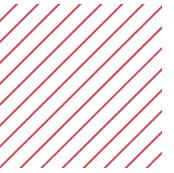
CD.yaml
on: workflow_dispatch

```
Deploy to staging      6s      Deploy to production  
PRODUCTION waiting for review
```

[Edit] [Delete] [+]

The screenshot shows a GitHub Actions pipeline interface. At the top, there's a summary card for a manual trigger: "Manually triggered 1 minute ago" by "Mikhail-M" (represented by a yellow sun icon) with commit "adcc885". The status is "Waiting", and total duration and artifacts are listed as "-". Below this is a yellow banner with a lightbulb icon and the message "Mikhail-M requested your review to deploy to PRODUCTION", with a "Review deployments" button. The main area displays the "CD.yaml" configuration for a "workflow_dispatch" event. It includes a "Deploy to staging" step (green checkmark icon) and a "Deploy to production" step (yellow clock icon). The production step is currently "waiting for review" and is associated with the "PRODUCTION" environment. At the bottom right, there are icons for editing, deleting, and adding steps.

DEMO:
Настраиваем CD с HELM



Стратегии деплоя

- recreate
- rolling-update
- blue/green
- canary



RECREATE

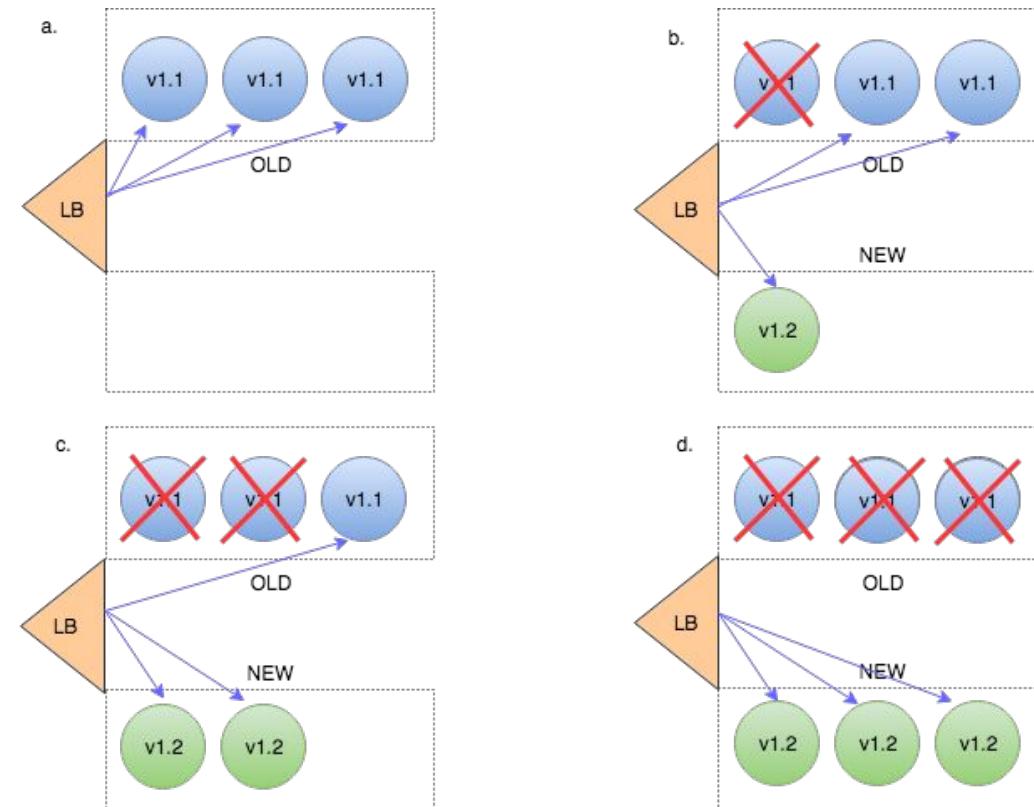
погасили версию А, поставили версию Б

- + лёгкая настройка
- + состояние приложения полностью обновлено
- downtime

ROLLING-UPDATE

версия Б медленно выкатывается в прод и заменяет версию А

- + лёгкая настройка
- + версия неспешно катится по всем инстансам
- может занять долгое время
- саппорт двух версий API
- плохой контроль трафика



Blue-Green

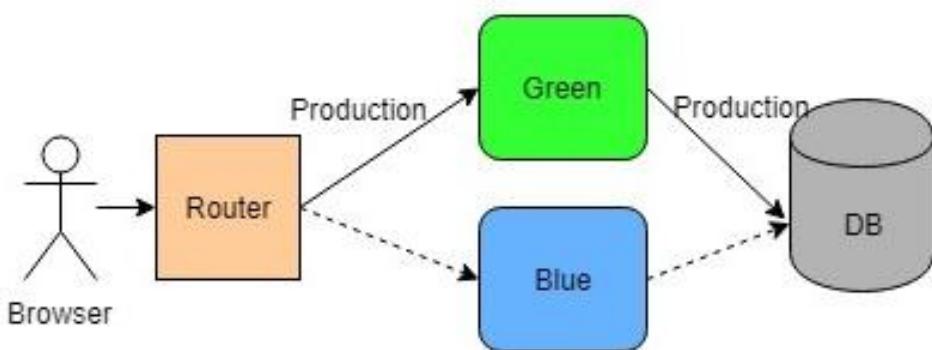
Включаем и старую и новую

модель

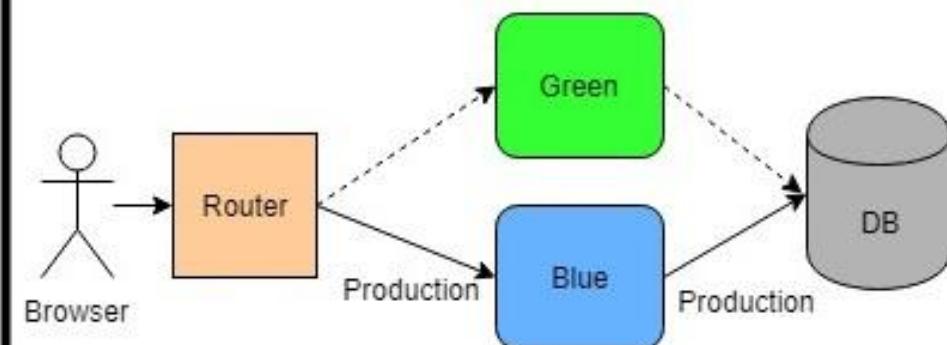
+ мгновенное переключение

- дорого и требует дублирования ресурсов

Before

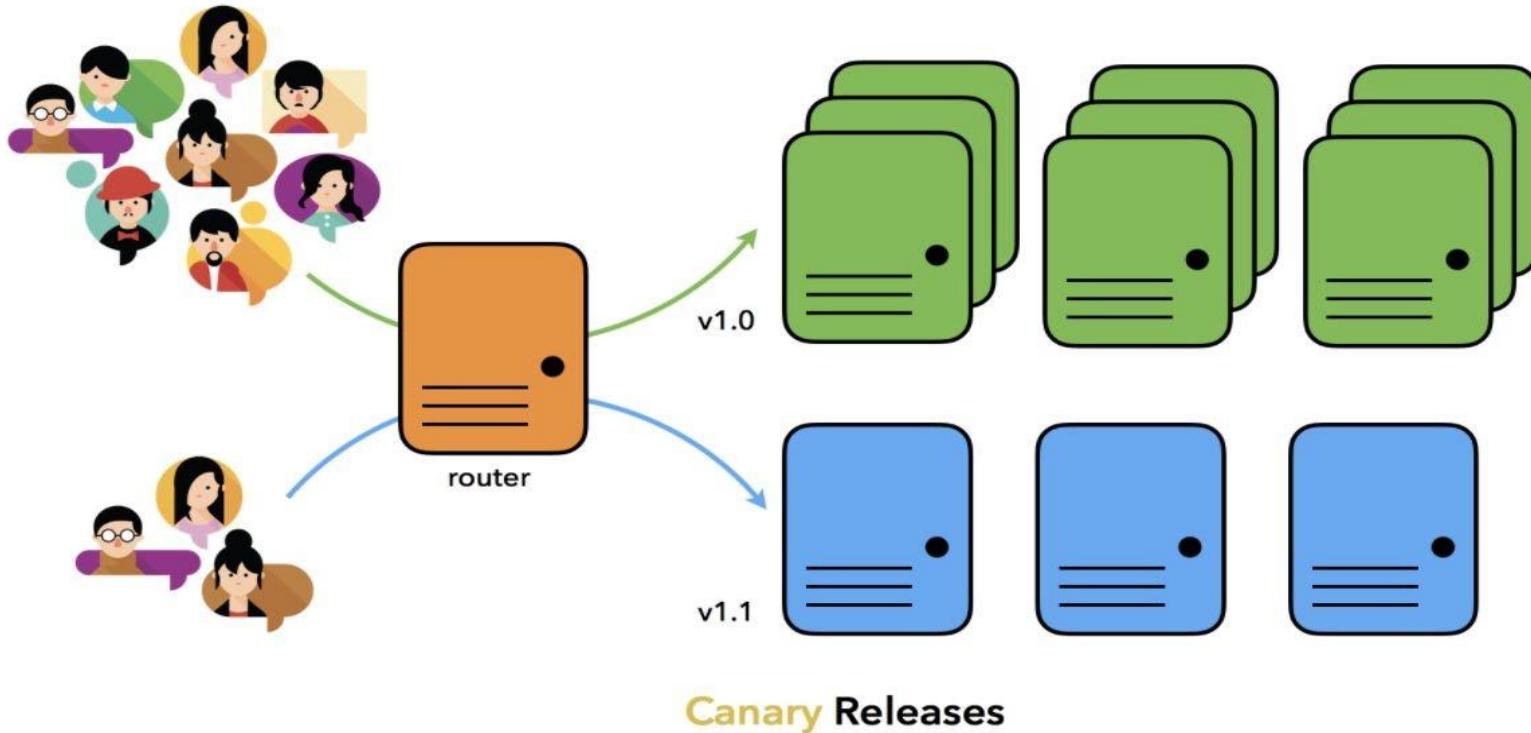


After



Canary

- выкатываем мало инстансов модели на часть пользователей и если что — сразу откатываем



DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.



Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	□□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	□□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■



Continuous Delivery vs Continuous Deployment

Continuous Deployment — это автоматический **выпуск и деплой** новых релизов.
Главное отличие от continuous delivery - отсутствие “кнопки” deploy

Опрос

<https://forms.gle/v2QrPtiT1QFEHfnY8>



академия
больших
данных



mail.ru
group

CI/CD для ML моделей

Михаил Марюфич, MLE

