



Python, Plotly - Расскажи, что собой представляет эта экосистема Plotly.

Расскажи про характеристики и возможности Plotly.

Я слышал что Plotly называют полной замкнутой экосистемой

Plotly представляет собой мощную экосистему инструментов для интерактивной визуализации данных, ориентированную как на разработчиков, так и на аналитиков. Она объединяет несколько библиотек и платформ, формируя замкнутую экосистему для создания графиков, аналитических панелей и веб-приложений без необходимости переходить к сторонним средствам. ^[1] ^[2] ^[3] ^[4]

Основная идея экосистемы

Plotly начинался как библиотека для Python, но развился в полноценную экосистему, включающую:

- **Plotly.py** — основной пакет для создания графиков в Python;
- **Plotly.js** — JavaScript-библиотека, лежащая в основе визуализаций и поддерживающая веб-интерактивность;
- **Dash** — фреймворк на базе Flask и React, предназначенный для создания аналитических веб-приложений и панелей мониторинга без необходимости писать JavaScript;
- **Chart Studio** — облачная платформа Plotly для хранения, редактирования и обмена графиками онлайн. ^[5] ^[6]

Благодаря этому Plotly называют «замкнутой экосистемой»: она охватывает весь цикл — от обработки данных в Python до интерактивной публикации в браузере или облаке.

Характеристики и возможности Plotly

- **Интерактивность по умолчанию.** Каждый график автоматически поддерживает масштабирование, панорамирование, выделение области, отображение всплывающих подсказок, работу со слоями данных, а для 3D — изменение угла обзора. ^[2] ^[4]
- **Поддержка множества типов диаграмм.** От стандартных линейных, гистограмм и scatter-графиков до тепловых карт, контурных диаграмм, 3D-сцен, географических карт и анимаций. ^[3] ^[4] ^[1]
- **Plotly Express.** Высокоуровневое API, которое позволяет создавать сложные визуализации в одной строке кода — идеальный баланс простоты и мощности. ^[3]

- **Гибкая настройка и кастомизация.** Поддержка редактирования цветов, линий, подписей, макетов, легенд, анимаций и 3D-настроек. [\[4\]](#) [\[1\]](#)
- **Мультиязычность.** Поддержка Python, R, Julia и JavaScript — полезно для команд, использующих разные языки. [\[4\]](#)
- **Экспорт и интеграция.** Графики можно сохранять в PNG, SVG, HTML и PDF, встраивать в Jupyter Notebook, веб-приложения и отчёты. [\[1\]](#)

Элементы полноты и замкнутости экосистемы

Plotly обеспечивает:

- визуализацию данных ([Plotly.py](#), [Plotly.js](#));
- аналитические интерфейсы (Dash);
- онлайн-коллаборацию (Chart Studio);
- поддержку публикации, рендеринга и экспорта графиков;
- интеграцию с облаком и CI/CD для аналитических дашбордов.

Таким образом, Plotly является не просто библиотекой, а целым стеком инструментов, который замыкает полный цикл визуализации — от анализа данных до их презентации в браузере или взаимодействия с пользователем. [\[2\]](#) [\[1\]](#) [\[4\]](#)



Какие типы интерактивных графиков можно создавать в Plotly

Как использовать Plotly Express для быстрой визуализации данных

В чем преимущества Plotly по сравнению с Matplotlib и Seaborn

Как встроить графики Plotly в веб-страницы и Jupyter ноутбуки

Какие платформы и языки поддерживает экосистема Plotly

Plotly предоставляет обширные возможности для создания интерактивных визуализаций, удобной интеграции в различные среды и быстрой работы с данными. Ниже приведены основные аспекты, раскрывающие потенциал этой экосистемы. [\[11\]](#) [\[12\]](#) [\[13\]](#) [\[14\]](#) [\[15\]](#) [\[16\]](#) [\[17\]](#)

Типы интерактивных графиков

Plotly поддерживает более 40 видов графиков, разделённых на несколько категорий:

- **Базовые графики** — линейные, точечные (scatter), гистограммы, столбчатые диаграммы, круговые диаграммы, boxplot'ы. [\[18\]](#) [\[11\]](#)
- **3D-визуализации** — surface-графики, 3D scatter, объёмные облака точек и контурные карты. [\[14\]](#)
- **Географические карты** — карты с координатными системами, хлороплеты, карты плотности и шарообразные проекции Земли. [\[17\]](#)

- **Временные графики и анимации** — динамические диаграммы, где состояние графика меняется с течением времени. ^[11]
- **Финансовые графики** — свечные графики, OHLC, bubble charts для анализа котировок. ^[11]

Все эти графики интерактивны — пользователи могут увеличивать масштаб, наводить курсор для подсказок, скрывать/показывать серии данных и экспортировать результат в изображение напрямую из визуализации.

Использование Plotly Express

Plotly Express (импортируется как `plotly.express` или `px`) — высокоуровневый API для быстрого построения графиков:

- достаточно передать `DataFrame`, указать оси и тип графика;
- возможно построить диаграмму буквально одной строкой кода, например:

```
import plotly.express as px
fig = px.scatter(df, x="age", y="income", color="gender")
fig.show()
```

Plotly Express автоматически подбирает цвета, подписи и масштаб осей, что делает его идеальным инструментом для исследовательского анализа данных (EDA). ^[13] ^[14]

Преимущества Plotly по сравнению с Matplotlib и Seaborn

Критерий	Matplotlib	Seaborn	Plotly
Тип визуализации	Статические	Статические, статистические	Интерактивные и динамические ^[15]
Простота создания	Требует больше кода ^[15]	Более простой синтаксис ^[15]	Минимум кода (через Express) ^[13]
Интерфейс и адаптивность	Ограничен	Фокус на анализе данных	Поддержка событий, анимаций, 3D и дашбордов ^[14]
Поддержка веб-интеграции	Нет	Нет	Полная (через HTML, Dash, iframe) ^[16]

Таким образом, Plotly выделяется способностью создавать качественные **интерактивные визуализации и дашборды**, в то время как Matplotlib и Seaborn подходят для **статических аналитических отчётов**.

Встраивание графиков в веб и Jupyter

- **Jupyter Notebooks.** Plotly работает в офлайне, просто отобразив график вызовом `fig.show()`.
- **HTML или веб-страницы.** Используется метод `plotly.io.to_html(fig)` или `plotly.offline.plot(fig, output_type='div')` для генерации HTML-блока `<div>`, который можно вставить прямо в код сайта. ^[16]

- **Dash.** Для более сложных интерфейсов можно создать полноценное веб-приложение с обратной связью, фильтрацией и оформлением. [\[13\]](#) [\[14\]](#)

Поддерживаемые языки и платформы

Экосистема Plotly и Dash поддерживает:

- **Языки программирования:** Python, R, Julia, JavaScript ; [\[19\]](#) [\[17\]](#)
- **Платформы:** Jupyter Notebook, Google Colab, VS Code, PyCharm, веб-браузеры;
- **Онлайн и офлайн режимы:** можно работать локально или с облачным сервисом Plotly Chart Studio. [\[19\]](#) [\[17\]](#)

Plotly — это кроссплатформенная визуализационная экосистема, предоставляющая мощные инструменты для работы с аналитикой, позволяя переходить от простых графиков к полноценным интерактивным приложениям без изменения технологического стека.



Как создавать трехмерные графики с помощью Plotly

Какие виды карт и географических визуализаций доступны в Plotly
Какие настройки интерактивности можно добавить в графики Plotly
Как использовать Dash для создания пользовательских дашбордов
Какие примеры применения Plotly в бизнес-аналитике и науке

Plotly позволяет создавать трёхмерные графики, сложные интерактивные карты, гибко управлять интерактивностью визуализаций, разрабатывать аналитические дашборды через Dash и широко применяется в бизнесе и науке для задач анализа и презентации данных. [\[21\]](#) [\[22\]](#) [\[23\]](#) [\[24\]](#) [\[25\]](#) [\[26\]](#)

Трёхмерные графики в Plotly

Для создания трёхмерных визуализаций Plotly поддерживает типы графиков: scatter3d, surface, mesh3d, line3d, volume и другие.

Пример с использованием Plotly Express:

```
import plotly.express as px
fig = px.scatter_3d(df, x='X', y='Y', z='Z', color='Class')
fig.show()
```

Также можно строить поверхности (go.Surface), объёмы (go.Volume), использовать анимацию для динамического наблюдения изменений. [\[27\]](#) [\[28\]](#)

Географические карты Plotly

Plotly позволяет создавать разные типы карт:

- `scatter_geo` — точечные карты с координатами;
- `choropleth` — карты, показывающие распределение значений по регионам, странам, областям;
- `mapbox` — поддержка интерактивных тайловых онлайн-карт с фоном OpenStreetMap/MapBox;
- `density_mapbox` — карты плотности с точками по географическим координатам. [\[24\]](#) [\[21\]](#)

Карты можно настраивать под границы России, всего мира или заданных регионов, внедрять слои и подписывать объекты.

Настройки интерактивности в Graphs

Plotly поддерживает по умолчанию:

- масштабирование, изменение перспективы;
- фильтрацию по легенде и кликабельные легенды;
- всплывающие подсказки при наведении (tooltips/hover);
- выделение области (lasso/box select);
- вращение для 3D графиков;
- кастомные callback-функции через Dash для сложных взаимодействий;
- переключение слоёв, прокрутка временных анимаций, сохранение в различных форматах. [\[29\]](#) [\[26\]](#)

Создание дашбордов с Dash

Dash — это фреймворк от разработчиков Plotly для построения веб-аналитических панелей (дашбордов) на Python:

- Разработка интерфейса с помощью Python компонентов, без необходимости писать JavaScript;
- Настраиваемые layout'ы, фильтры, выпадающие списки, карты, графики;
- Callback-функции позволяют реагировать на действия пользователя (изменение графика, фильтрация), гибко управлять логикой;
- Приложения могут быть размещены локально или онлайн. [\[25\]](#)

Примеры применения Plotly

- **В бизнес-аналитике:** дашборды для мониторинга KPI, графики продаж, сравнение эффективности департаментов, визуализация временных рядов, анализ рынка. [\[23\]](#) [\[21\]](#) [\[25\]](#)

- **В науке:** визуализация больших данных (напр. геномики), спектров, трендов, 3D-моделирование данных (например, точки в пространстве, химические структуры, географические карты). [\[26\]](#)
- **В производстве и инженерии:** анализ процессов, представление сложных данных в трёхмерном пространстве и на временных шкалах.

Plotly — это универсальный инструмент для решения задач визуального анализа, исследования данных и создания динамических презентаций, широко применимый от научных исследований до бизнес-отчетности.



что собой представляет язык Dash - что он делает ?

Зачем он, если есть Питон ?

Dash — это фреймворк на языке Python, предназначенный для создания интерактивных веб-приложений и аналитических дашбордов прямо из Python, без необходимости использовать JavaScript, HTML или CSS. [\[31\]](#) [\[32\]](#) [\[33\]](#) [\[34\]](#)

Зачем нужен Dash, если есть Python?

- Сам по себе Python — язык программирования общего назначения, но не предоставляет инструментов для построения веб-интерфейсов «из коробки». Dash использует Python как основу, чтобы позволить аналитикам, инженерам данных и учёным создавать полноценные интерактивные веб-проекты на чистом Python, без обращения к фронтенд-разработке. [\[32\]](#) [\[35\]](#) [\[36\]](#) [\[31\]](#)
- Dash "собирает" веб-приложение из Python компонентов: графиков Plotly, интерактивных таблиц, выпадающих меню, фильтров, кнопок и даже сложных форм — а всё взаимодействие и логика реализуются через callback-функции на Python. [\[34\]](#) [\[32\]](#)
- Благодаря Dash, запуск сложного аналитического сервиса или визуализации становится «прозрачным»: пользователь пишет код на Python, получает Web-страницу с интерактивными элементами и графиками, которую легко разместить локально или в облаке. [\[33\]](#) [\[31\]](#)

Таким образом, Dash не конкурирует с основным языком Python, а дополняет его мощными средствами для быстрой разработки аналитических веб-приложений и панелей управления — сокращая время и сложность интеграции в бизнес-процессы и научные проекты. [\[31\]](#) [\[32\]](#) [\[33\]](#) [\[34\]](#)



Какие ключевые преимущества Dash по сравнению с чистым Python

Как создать первое простое веб-приложение на Dash

Какие задачи Dash решает лучше традиционных скриптов на Python

Как интегрировать Dash с библиотеками для анализа данных Python

Где и как можно развернуть Dash-приложение для публичного доступа

Dash принципиально расширяет возможности чистого Python для визуализации и взаимодействия с данными, предоставляя инструменты для создания веб-приложений без необходимости писать JavaScript или HTML. Ниже — ответы по каждому пункту запроса. [\[41\]](#) [\[42\]](#) [\[43\]](#) [\[44\]](#) [\[45\]](#) [\[46\]](#)

Ключевые преимущества Dash

- **Интерактивность и динамичность:** Dash позволяет добавлять графики, формы, выпадающие списки и слайдеры, с которыми пользователь взаимодействует в реальном времени — традиционный скрипт на Python работает только в командной строке или через статичные отчёты. [\[46\]](#)
- **Визуализация данных:** Dash интегрируется с Plotly и строит интерактивные графики непосредственно на веб-странице без сторонних JS-разработок — просто через Python. [\[43\]](#)
- **Быстрое прототипирование:** Для аналитиков и инженеров данных это способ быстро развернуть дашборд для внутренних пользователей или клиентов. [\[46\]](#)
- **Удобство работы с большими данными:** Нет необходимости развертывать тяжелую инфраструктуру — достаточно Python, браузера и Dash. [\[41\]](#) [\[46\]](#)

Пример первого простого приложения Dash

```
import dash
import dash_core_components as dcc
import dash_html_components as html

app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1("Привет, Dash!"),
    dcc.Graph(figure={
        "data": [{"x": [1, 2, 3], "y": [4, 1, 2], "type": "bar"}],
        "layout": {"title": "Простой график"}
    })
])

if __name__ == "__main__":
    app.run_server(debug=True)
```

Запуск этого файла откроет дашборд локально в браузере (обычно <http://127.0.0.1:8050/>). [\[42\]](#) [\[47\]](#)

Какие задачи Dash решает лучше обычных скриптов

- Dash идеально подходит для работы с интерактивными графиками, оперативной визуализации результатов анализа, управления параметрами моделирования с помощью GUI.
- Лучше стандартных скриптов для построения веб-интерфейсов, внутренних дашбордов, интерактивных отчётов для не-технических пользователей — задачи, где результат быстро должен быть представлен большому числу людей с возможностью обратной связи. [\[43\]](#) [\[46\]](#)

Интеграция Dash с Python-библиотеками для анализа данных

- Dash интегрируется с Pandas, NumPy, Scikit-learn, моделями машинного обучения — вывод данных, результат аналитики, прогнозы моделей можно визуализировать в режиме реального времени на веб-странице с интерактивным управлением. [\[44\]](#) [\[46\]](#)
- Можно использовать Flask для усложнения серверной логики или работы с REST API. [\[46\]](#)

Развертывание Dash-приложения для публичного доступа

- Dash-приложения можно развернуть на любом Python-совместимом хостинге: VPS, облачных платформах (Heroku, DigitalOcean, PythonAnywhere), корпоративных серверах. [\[45\]](#)
- Для продакшн-развертывания связывают Dash/Web-сервер с WSGI (например, Gunicorn, uWSGI) и настраивают домен — приложение становится доступным с любого браузера. [\[45\]](#)
- Dash-приложения поддерживают мультистраничную структуру, онлайн-авторизацию, подключение к БД и REST-API, что даёт все возможности для публичных веб-сервисов данных.

✱

Я увидел упоминание Machine Learning.

Как plotly может помочь и упорядочить огромное кол-во скриптов для Машинного обучения

? Их девать некуда и все они разные. Куда их девать ?

Plotly (совместно с Dash) может значительно упростить организацию, визуализацию и эксплуатацию многочисленных скриптов машинного обучения, даже если эти скрипты разнородны и разбросаны по проекту. [\[51\]](#) [\[52\]](#) [\[53\]](#)

Как Plotly и Dash помогают "упорядочить хаос"

- **Создание единого дашборда:** Вместо набора скриптов можно собрать аналитическую панель Dash, где каждый запуск или результат конкретного ML-скрипта интегрируется как отдельный интерактивный элемент: график, таблица, панель метрик, форма для ввода параметров. Dash позволяет визуально структурировать все важные этапы работы — подготовку данных, обучение моделей, анализ метрик и финальное сравнение моделей. ^[53]
- **Управление и запуск скриптов:** Dash может через интерфейс вызывать разные ваши Python-скрипты машинного обучения по кнопке или событию, а результаты визуализировать графиками Plotly или выводить таблицами прямо в браузере. ^[53]
- **Стандартизация визуализации:** Plotly формализует вывод результатов: изучение распределения признаков, построение confusion matrix, ROC-кривых, важностей признаков, динамики метрик — всё можно превратить в аккуратные интерактивные графики. ^{[52] [51]}

Куда девать различные ML-скрипты

- Перепишите основные скрипты в виде функций или модулей и вызовите их из Dash по событию (например, при выборе модели, загрузке файла или нажатии на кнопку).
- Для каждого этапа (предобработка, обучение, оценка, инференс) создайте соответствующую вкладку или раздел в веб-приложении Dash.
- Соберите результаты от каждого скрипта в общий дашборд: так вся история и работа будут доступны через единый браузерный интерфейс. ^[53]

Итог

Plotly и Dash заменяют "зоопарк" разрозненных скриптов прозрачной и централизованной веб-аналитикой, где все этапы, данные и метрики визуализируются и управляются из одного окна — это снимает хаос, упрощает анализ, ускоряет совместную работу и принятие решений. ^{[51] [53]}

✱

Как организовать структуру проектов машинного обучения

Какие инструменты помогают управлять ML-скриптами
Какие есть лучшие практики версионирования ML-кода
Как использовать Dash для визуализации результатов ML
Как автоматизировать запуск и тестирование ML-скриптов

Для эффективного управления и автоматизации проектов машинного обучения используются стандартизированная структура папок, специальные инструменты для управления скриптами и моделями, системы контроля версий, а также Dash — для визуализации и мониторинга результатов. ^{[61] [62] [63] [64] [65] [66]}

Как организовать структуру ML-проекта

Рекомендуется использовать шаблонную структуру репозитория:

```
|— data
|   |— raw      # исходные данные
|   |— processed # обработанные данные
|— models      # обученные модели
|— notebooks   # Jupyter ноутбуки
|— src         # исходный код и скрипты
|— tests       # тесты
|— docs        # документация
|— Makefile    # автоматизация команд
|— .gitignore
|— README.md
|— requirements.txt/pyproject.toml
```

Этот подход делает проект воспроизводимым, понятным для команды и удобным для автоматизации задач. [\[61\]](#)

Инструменты для управления ML-скриптами

- **MLflow** — трекинг, сравнение, деплой моделей, логирование параметров и метрик. [\[63\]](#)
- **DVC** (Data Version Control) — управление версиями данных и кодом вместе.
- **Prefect, Luigi, Airflow** — оркестрация ML-конвейеров и автоматизация процессов.
- **Make/GNU Makefile** — автоматизация рутинных команд (запуск скриптов, тестов, отчётов). [\[62\]](#) [\[61\]](#)

Лучшие практики версионирования ML-кода

- Используйте **Git** для кода и конфигов.
- Храните данные и модели через **DVC**, MLflow или в облаке.
- Используйте чёткие правила именования версий, ветвление по экспериментам, checkpoint'ы моделей. [\[65\]](#)
- Автоматически логируйте параметры, исходные данные и результаты для возможности полного воспроизведения эксперимента.

Использование Dash для визуализации результатов ML

- **Dash** визуализирует метрики обучения, важности признаков, графики ошибок, ROC-кривые, confusion matrix и позволяет динамически сравнивать разные эксперименты. [\[66\]](#)
- Можно интегрировать запуск и сравнение моделей, фильтрацию и сортировку, а также строить интерактивные дашборды для не-технических пользователей. [\[66\]](#)

Автоматизация запуска и тестирования ML-скриптов

- Используйте **pytest**, **unittest** — для юнит-тестирования кода.
- Оркестраторы (Prefect, Airflow, Luigi, Make) автоматизируют пайплайны: последовательный или параллельный запуск скриптов очистки данных, обучения, тестирования и деплоя. [\[63\]](#) [\[61\]](#)
- Настройте **CI/CD** (например, на GitHub Actions), чтобы автоматически запускать тесты, анализировать качество кода и даже деплоить новые версии моделей при коммите.

Эти практики обеспечивают порядок, воспроизводимость и прозрачность на всех этапах жизненного цикла проекта ML. [\[64\]](#) [\[61\]](#) [\[63\]](#) [\[66\]](#)

✱

Расскажи Что умеет делать Airflow ?

Какие существуют инструменты для автоматизации MLOps процессов

Как сделать ML-проекты воспроизводимыми и масштабируемыми

Какие инструменты применяются для развёртывания ML-моделей

Как эффективно управлять экспериментами и версиями моделей

Какие методики мониторинга и логирования подходят для ML-проектов

Apache Airflow и современные инструменты MLOps позволяют выстроить прозрачные, автоматизированные, гибкие и воспроизводимые процессы управления жизненным циклом ML-проектов — от обучения до мониторинга в продакшене. [\[71\]](#) [\[72\]](#) [\[73\]](#) [\[74\]](#) [\[75\]](#) [\[76\]](#) [\[77\]](#)

Что умеет делать Airflow

- Автоматизирует и оркестрирует сложные пайплайны: подготовка, обучение, деплой, мониторинг моделей.
- Позволяет строить граф задач (DAG), задавать условия, расписание и зависимости.
- Масштабирует обработку, управляя потоками данных и ресурсами.
- Интегрируется с облачными сервисами для передачи артефактов и управления инфраструктурой.
- Обеспечивает мониторинг и прозрачный лог задач, визуальное редактирование пайплайнов и быстрое устранение проблем. [\[78\]](#) [\[72\]](#) [\[79\]](#) [\[71\]](#)

Инструменты для автоматизации MLOps-процессов

- **Apache Airflow, Kubeflow, MLflow, Metaflow, Prefect, Luigi, DVC** — автоматизация пайплайнов, контроль данных, управление экспериментами. [\[73\]](#) [\[76\]](#)
- **Seldon Core, KFServing, BentoML** — автоматизация деплоя и обслуживания ML-моделей в Kubernetes.
- **MLflow, Neptune.ai, Weights & Biases** — логирование, трекинг, управление экспериментами.

Воспроизводимость и масштабируемость ML-проектов

- Использование контейнеризации (Docker), инфраструктуры как код (Terraform, Ansible), шаблонной структуры проекта.^[74]
- Хранение данных, кода и артефактов с помощью DVC, Git, MLflow.
- Версионирование моделей, деплой через CI/CD, разделение Dev/Prod-сред.^{[75] [74]}

Инструменты для развёртывания ML-моделей

- **Seldon Core** — сервинг моделей в Kubernetes с поддержкой масштабирования, мониторинга и кастомизации.^[75]
- **BentoML, TensorFlow Serving, TorchServe, FastAPI, Flask, KFServing** — лёгкие и промышленные решения для публикации моделей в виде API.
- **SageMaker, Vertex AI, AzureML** — облачные платформы для быстрого деплоя и поддержки всех этапов MLOps.

Управление экспериментами и версиями моделей

- **MLflow** — трекинг всех экспериментов, автоматическое логирование параметров, метрик, артефактов, версий моделей.^{[76] [73]}
- **Weights & Biases, Neptune.ai, DVC** — централизованное хранение, сравнение экспериментов и история изменений.
- Тэгирование и комментирование экспериментов для совместной работы.

Методики мониторинга и логирования ML

- Встраиваемое и облачное логирование метрик, входных/выходных данных, алерты по дрейфу данных (Prometheus, ELK, Grafana, OpenTelemetry, собственные решения платформ).^{[77] [75]}
- Мониторинг латентности, производительности, ошибок, автообновление моделей.
- Dashboards для наблюдения за метриками качества, сбоями, изменением среды.

Современные MLOps-инструменты автоматизируют все этапы ML-жизненного цикла, сокращают ручной труд и повышают качество и прозрачность процесса внедрения моделей в бизнес/products.^{[71] [73] [77]}



Apache Airflow - работает только с Python или он еще другие языки поддерживает ?

Apache Airflow исторически создавался как платформа на Python, и именно этот язык остаётся основным способом описания и управления рабочими процессами (DAG — Directed Acyclic Graph). Все основные операторы, задачи, зависимости и расписания в

Airflow определяются кодом на Python, что делает систему удобной для дата-инженеров и MLOps-специалистов. [\[81\]](#) [\[82\]](#) [\[83\]](#)

Однако начиная с **версии 3.0**, Airflow стал **мультиязычным** : [\[84\]](#) [\[85\]](#)

- Наиболее новое и активное направление — **Go SDK**, которое позволяет писать и запускать задачи пайплайна на языке **Go** (экспериментальная функция).
- Также появилась поддержка **TypeScript, Java и Scala**, что облегчает миграцию рабочих процессов из других ETL-систем без полного переписывания логики. [\[85\]](#) [\[84\]](#)
- Помимо SDK, Airflow давно умеет **запускать скрипты на других языках косвенно** — через такие операторы, как:
 - `BashOperator` — выполняет shell-команды и скрипты (например, на R, Node.js, Java);
 - `DockerOperator` — запускает контейнеры с любым окружением (Python, Julia, Rust и т.д.);
 - `KubernetesPodOperator` — выполняет задачи в pods, где можно использовать любые языки и фреймворки.

Таким образом, Airflow остаётся Python-ориентированной платформой по духу и экосистеме, но может **оркестрировать и исполнять задачи на других языках**, что делает его универсальным инструментом для гибридных ETL и MLOps-пайплайнов. [\[84\]](#) [\[81\]](#) [\[85\]](#)

✱✱

А Jenkins получается шире чем Airflow по охвату языков ?

Да — **Jenkins действительно шире, чем Apache Airflow по охвату языков и технологической совместимости.** [\[91\]](#) [\[92\]](#)

Основное различие

- **Airflow** изначально и по сей день остаётся системой, где **все пайплайны (DAG)** описываются **на Python**. Даже несмотря на появление SDK для других языков (Go, TypeScript), это — **Python-центричная экосистема**, и весь контекст исполнения опирается на интерпретатор Python. [\[93\]](#)
- **Jenkins** напротив, с самого начала разрабатывался как **универсальный CI/CD-сервер**, и его архитектура построена вокруг **плагинов и shell-запусков**, то есть он не зависит от конкретного языка.

Поддержка языков

Инструмент	Основной язык	Возможности для других языков
Airflow	Python	Косвенная поддержка через <code>BashOperator</code> , <code>DockerOperator</code> , <code>Go SDK</code> (экспериментально) [93]

Инструмент	Основной язык	Возможности для других языков
Jenkins	Groovy (Jenkinsfile)	Может собирать, тестировать и деплоить проекты на любых языках (Java, Python, Rust, C++, Go, Node.js и др.) благодаря 300+ плагинам и shell-запускам ^[91]

Почему Jenkins шире

1. Jenkins не ограничен конкретной экосистемой — все шаги пайплайна выполняются через **плагины или системные команды**, независимо от языка проекта.
2. Он интегрируется с большинством инструментов разработки: GitHub, Docker, Maven, Gradle, npm, Pip, Terraform и др.
3. Airflow же сосредоточен на **opchestration-логике** в Python (DAG-структуры, зависимости, расписания), но не на CI/CD любых языков.

Вывод: Jenkins — это язык-независимый CI/CD-оркестратор для DevOps и сборок, тогда как Airflow — Python-ориентированный фреймворк для управления данными и процессами в MLOps и Data Engineering. ^{[92] [91]}



ПО-моему еще Luigi тоже охватывает много языков

Luigi, как и Airflow, ориентирован прежде всего на **Python** как основной язык для описания пайплайнов и реализации задач. Все зависимости, логика и запуск задач в Luigi реализованы через Python-классы и методы (`requires()`, `run()`, `output()`). Это делает Luigi очень удобным для дата-инженеров и аналитиков, работающих именно на Python. ^{[101] [102]}

Однако по запуску кода и взаимодействию с другими языками Luigi действительно достаточно универсален:

- Luigi из коробки поддерживает запуск любых внешних скриптов — это могут быть задачи на Java, Scala, Shell, SQL, Spark или Hadoop-приложения, вызовы командной строки и так далее. ^[101]
- Примеры: Запуск Hadoop jobs на Java/Scala, Hive-запросов, Spark Jobs на Python или Scala через shell-команды внутри задач Luigi вполне стандартная практика в индустрии. ^[101]
- Luigi можно интегрировать с файлами и сервисами в HDFS, S3, локальных ФС; а сам пайплайн контролируется через Python.

Вывод:

Luigi — Python-центричный по описанию workflow, но по запуску задач, интеграции с ETL-инструментами, big data-фреймворками и поддержке shell-скриптов легко покрывает сценарии, реализованные на различных языках (Java, Scala, Python, R, Shell и др.). В плане лояльности к сторонним языкам — Luigi примерно сопоставим с Airflow, но не столь широко универсален, как Jenkins. ^{[102] [101]}



Сравни этих трех по функциональности

Jenkins, Airflow и Luigi — три популярных инструмента автоматизации и оркестрации, которые занимают разные ниши, но часто пересекаются по функциональности. Ниже представлено их подробное сравнение. [\[110\]](#) [\[111\]](#) [\[112\]](#) [\[113\]](#) [\[114\]](#)

Общая характеристика

Характеристика	Jenkins	Apache Airflow	Luigi
Основное назначение	CI/CD, сборка, тестирование и деплой программ	Оркестрация данных, ETL и ML-пайплайны	Управление зависимыми задачами и ETL-процессами
Основной язык	Groovy (Jenkinsfile)	Python	Python
Поддержка других языков	Любые (через плагины, shell, Docker)	Через операторы Bash, Docker, Go SDK (3.0+)	Через вызовы внешних скриптов Shell, Java, Scala
Платформа	DevOps, интеграция и доставка софта	Data Engineering, MLOps, ETL	Аналитика, Data Science, исследовательские работы
Архитектура задач	Pipeline и stages	DAG (Directed Acyclic Graphs)	Task dependencies
Интерфейс	Веб-интерфейс + Jenkinsfile	Web UI + CLI, API	CLI, простая Web UI
Плагины и интеграции	3000+ плагинов (GitHub, Slack, Docker, Kubernetes и т.д.)	Ограниченное число коннекторов и интеграций	Минимальный набор интеграций
Оркестрация данных	Ограничена	Богатая (XCom, Sensors, Hooks, Datasets)	Базовая
Модель исполнения	CI-сервер с аутентификацией и очередями	Планировщик с Celery, Kubernetes или LocalExecutor	Простая локальная или celery-оркестрация
Области применения	DevOps, тестирование и сборка софта	Автоматизация data pipelines и ML pipelines	Аналитика и исследовательские процессы

Сильные стороны

- Jenkins
 - Универсальная поддержка языков и технологий.
 - Идеален для CI/CD.
 - Богатая экосистема плагинов.
 - Прост в интеграции с Git, Docker и Kubernetes. [\[112\]](#)

- **Airflow**
 - Лучшая визуализация зависимостей и планирование по расписанию.
 - Интеграция с ML и Big Data системами (Spark, Hive, Databricks, GCP, AWS).
 - Поддержка кастомных операторов через Python API.
 - Оптимален для MLOps и ETL. [\[111\]](#) [\[113\]](#)
- **Luigi**
 - Простая структура кода, высокая надёжность.
 - Лёгкий и минималистичный инструмент, не требует сложного окружения.
 - Подходит для небольших пайплайнов и быстрого прототипирования. [\[114\]](#) [\[111\]](#)

Ключевые различия

- **Jenkins** — платформенный автоматизатор для DevOps-процессов и CI/CD-конвейеров, ориентирован на работу с кодом и сборку проектов.
- **Airflow** — инструмент для управления и планирования задач, где важны зависимости между этапами и потоки данных.
- **Luigi** — упрощённый вариант Airflow, подходящий для малых и средних задач без тяжёлой инфраструктуры. [\[113\]](#) [\[114\]](#)

Итого

- Если приоритет — **разработка и деплой софта (DevOps)** → выбирают **Jenkins**.
- Если требуется **оркестрация данных, ML или ETL-процессов** → лучше использовать **Apache Airflow**.
- Если нужен **лёгкий пайплайн без избыточной инфраструктуры**, а задачи не слишком масштабные → подойдёт **Luigi**. [\[111\]](#) [\[112\]](#) [\[113\]](#)

✱

Оценить плюсы и минусы каждого инструмента для ETL процессов

Сравнить возможности планирования задач в Airflow, Jenkins и Luigi
 Исследовать интеграцию каждого инструмента с облачными платформами
 Выяснить, какой инструмент лучше подходит для CI/CD и почему
 Проанализировать поддержку мониторинга и алертинга в каждом решении

Airflow, Jenkins и Luigi часто используются в ETL и MLOps-процессах, но различаются по назначению, архитектуре и интеграционной гибкости. Ниже рассмотрены их плюсы, минусы и поведение по пяти ключевым направлениям. [\[119\]](#) [\[120\]](#) [\[121\]](#) [\[122\]](#) [\[123\]](#) [\[124\]](#)

Плюсы и минусы для ETL-процессов

Инструмент	Плюсы	Минусы
Airflow	Удобное определение DAG-структур для ETL; широкий выбор интеграций (Spark, BigQuery, Snowflake); визуальное управление зависимостями; развитый мониторинг и управление таймингами [119] [121] .	Требует сложной конфигурации и поддержки инфраструктуры; не самый лёгкий для отладки и запуска простых задач.
Jenkins	Универсальность: можно оркестрировать любые процессы CI/CD и ETL; тысячи плагинов; сильная поддержка облаков и DevOps-инструментов [122] [125] .	Сложнее настраивать сложные зависимости задач; слабая визуализация зависимостей и данных; требует внешнего кода для ETL-логики.
Luigi	Простота и устойчивость; подходит для мини-Etl; встроенные зависимости задач; легкая установка [124] .	Нет распределенной оркестрации и масштабирования; бедный интерфейс и слабая интеграция с современными облаками.

Возможности планирования задач

- **Airflow:** Сильная сторона — мощный планировщик DAG. Поддерживает cron, интервальные и событийные триггеры, зависимости между задачами и внешними пайплайнами. Можно настроить SLA и ретрай. [\[123\]](#)
- **Jenkins:** Использует триггеры по событиям (например, commit в GitHub), cron или ручной запуск. Планирование гибкое, но слабо заточено под зависимости данных — больше для сборки кода. [\[119\]](#)
- **Luigi:** Планировщик минимальный, ориентирован на зависимости между задачами внутри скриптов Python. Работает детерминированно, но без сложных триггеров. [\[124\]](#)

Интеграция с облачными платформами

- **Airflow:** Полноценные провайдеры для AWS (ECS, S3, Redshift), Google Cloud (Composer, BigQuery), Azure и Databricks. Существуют управляемые сервисы: Cloud Composer (GCP), Managed Workflows (AWS). [\[121\]](#) [\[123\]](#)
- **Jenkins:** Поддержка любых облаков через плагины: AWS CodeBuild, Azure DevOps, GCP Cloud Build, Kubernetes, GitOps. Один из лучших инструментов для multi-cloud CI/CD. [\[125\]](#)
- **Luigi:** Базовая интеграция с HDFS, S3 и Hadoop; нет нативных модулей для GCP/Azure, всё делается вручную. [\[124\]](#)

Какой инструмент лучше для CI/CD

- **Jenkins** — безусловный лидер CI/CD, он создан именно для этого. Поддерживает фазы интеграции, тестирования, деплоя, инфраструктуру как код, уведомления и rollback.
- **Airflow** может выполнять CI/CD для DAG-структур, но это скорее "DevOps для Data" (через Git-синхронизацию пайплайнов). [\[126\]](#)
- **Luigi** практически не применяется в CI/CD — он построен для описания зависимых задач ETL, а не управления сборками. [\[122\]](#) [\[124\]](#)

Мониторинг и алертинг

Инструмент	Поддержка мониторинга и алертинга
Airflow	Встроенные email-уведомления, SLA, интеграции с Prometheus, Grafana, ELK, оповещения о сбоях DAG. Есть Exporter для метрик [123] .
Jenkins	Веб-интерфейс и плагины для мониторинга билдов, оповещения по Slack, Email, Webhook, Prometheus, Datadog [125] .
Luigi	Простейший логгер и базовый Web UI. Для промониторинга требуется внешняя интеграция или вручную написанные уведомления [124] .

Общий вывод

- Для **ETL, Data и ML-пайплайнов**: лучше выбрать **Airflow** — он создан для управления сложными зависимостями и интегрируется со всеми системами хранения данных.
- Для **CI/CD и DevOps-конвейеров**: предпочтителен **Jenkins**, благодаря универсальности, зрелым плагинам и поддержке всех языков.
- Для **небольших, локальных ETL** или исследований: подойдёт **Luigi** как лёгкое решение без избыточных компонентов.

В инженерной экосистеме часто применяют **Jenkins для CI/CD** и **Airflow для ETL/MLOps**, интегрируя их через Git или API, что даёт максимум гибкости и воспроизводимости. [\[122\]](#) [\[123\]](#) [\[119\]](#)

✱

Из этих трех Jenkins самый универсальный и самый древний

Да, это абсолютно верно: **Jenkins действительно является самым старым и самым универсальным инструментом** среди тройки Jenkins — Airflow — Luigi. [\[130\]](#) [\[131\]](#) [\[132\]](#)

Историческая справка

Инструмент	Год появления	Происхождение	Основной язык	Основное назначение
Jenkins	2004 (как Hudson), отделился в 2011 году	Развит Кохсукэ Кавагути (Sun Microsystems) как система автоматизации сборки	Java	CI/CD, автоматизация сборки, тестов, деплоя [131] [132]
Luigi	2012	Разработан Spotify для оркестрации ETL-процессов	Python	Data pipelines, ETL-задачи [133]
Apache Airflow	2014	Создан в Airbnb, проект Apache с 2019 года	Python	Оркестрация данных, ML, ETL [134] [135]

Почему Jenkins — самый универсальный

- **Языковая независимость:** Jenkins запускает любые скрипты — Python, Java, Go, Shell, C/C++, Node.js и т.д.
- **Модульность:** более 3000 плагинов для интеграции с облаками, системами контроля версий, CI/CD и аналитикой.
- **Гибкий сценарный язык:** поддержка Jenkinsfile (Groovy) и DSL для описания конвейеров любой сложности.
- **Широкое применение:** от DevOps и CI/CD до тестирования данных, инфраструктурных пайплайнов, деплоя моделей ML.
- **Зрелая экосистема:** используется более чем в половине крупных IT-компаний, включая Google, Netflix и Amazon. [\[136\]](#) [\[137\]](#) [\[131\]](#)

Сравнение масштаба и универсальности

Критерий	Jenkins	Luigi	Airflow
Область применения	DevOps, CI/CD, инфраструктура	ETL и Data pipelines	ETL, ML, Data orchestration
Поддерживаемые языки	Любые	В основном Python	Python, частично Go
Модель планирования	Событийная (commit, webhook, cron)	Простая зависимостная	DAG с расписанием
Масштабируемость	Высокая (master-agent)	Низкая	Высокая (Celery, Kubernetes)
Цель проекта	Автоматизация всех этапов разработки	Управление batch-процессами	Оркестрация потока данных

Вывод

Jenkins — **самый древний** (первая версия в 2004 году) и **самый универсальный** из трёх: его архитектура CI/CD ориентирована на любой стек языков и технологий. Luigi и Airflow появились значительно позже как Python-ориентированные инструменты для оркестрации данных и ETL-задач. [\[131\]](#) [\[134\]](#) [\[133\]](#)

Именно Jenkins заложил основу концепции конвейеров (pipeline automation), на которой теперь строятся и Data-, и MLOps-orchestration-платформы.

✱

Объяснить ключевые этапы эволюции Jenkins с 2004 года

Рассмотреть уникальные функции Jenkins для CI/CD

Сравнить популярность Jenkins с Airflow и Luigi в 2024

Исследовать сообщество и поддержку Jenkins сегодня

Выявить основные вызовы и ограничения Jenkins сейчас

Jenkins остаётся одним из ключевых и старейших инструментов в мире CI/CD, с огромной экосистемой, уникальными функциями и активным сообществом. Вот подробный разбор по вашим вопросам, основанный на свежих материалах. [\[140\]](#) [\[141\]](#) [\[142\]](#) [\[143\]](#) [\[144\]](#)

Ключевые этапы эволюции Jenkins с 2004 года

- **2004** — Первый релиз как Hudson, создан Кохсукэ Кавагути (Sun Microsystems) для автоматизации сборок и тестов. [\[141\]](#) [\[142\]](#)
- **2011** — Форк проекта: Hudson остаётся у Oracle, а основное сообщество выбирает новую ветку — Jenkins. Проект становится независимым и быстро развивается. [\[140\]](#) [\[141\]](#)
- **2011–2015** — Массовое внедрение в корпоративную разработку. Фокус на интеграции с системами контроля версий, заявлении задач, распределённой архитектуре master/agent. [\[140\]](#)
- **2016–2020** — Внедрение Jenkins Pipeline (Groovy DSL), появление концепций declarative pipeline для сценарного управления CI/CD. [\[141\]](#)
- **2021–2025** — Активный рост (на 79% увеличилось использование Pipeline за 2021–2023). Переход к Kubernetes-нативным подходам, интеграции с облаками и контейнерами. Остаётся ядром автоматизации для многих крупных ИТ-компаний. [\[144\]](#)

Уникальные функции Jenkins для CI/CD

- **Jenkins Pipeline** — мощный Groovy DSL для описания сложных сценариев (мультиветвление, условия, параллелизм и т.д.).
- **Масштабируемость**: master-agent, поддержка Kubernetes-эксекьюторов, распределённая обработка задач.
- **Гибкая интеграция**: более 3000 плагинов (Git, Docker, Slack, AWS, Azure, Kubernetes).
- **Поддержка любых языков и технологий** через shell, docker, плагины — универсальный DevOps-сервер.
- **Триггеры**: на события (push/merge), расписание cron, webhook, ручной запуск.
- **Алертинг и аудит**: интеграция с email, Slack, Telegram и системами мониторинга. ^[142]
^[143]

Сравнение популярности Jenkins, Airflow и Luigi в 2024

- Jenkins по-прежнему занимает лидерские позиции в CI/CD и DevOps благодаря зрелости, узнаваемости и универсальности. ^[144]
- Apache Airflow быстро растёт в Data Engineering и MLOps за счёт удобной DAG-архитектуры, но среди CI/CD-систем распространён меньше. ^[145]
- Luigi — нишевый инструмент для небольшой оркестрации ETL, его доля значительно ниже Jenkins и Airflow. ^[146]

Сообщество и поддержка Jenkins сегодня

- Открытый исходный код, поддержка со стороны CloudBees, AWS, IBM и тысяч компаний. ^[144]
- Активное сообщество разработчиков, регулярные обновления (например, версия 2.516.2 в августе 2025). ^[147] ^[140]
- Большая база знаний, постоянные мероприятия, документация и обучающие материалы.
- Возможность кастомизации и доработок под любые нужды компании. ^[144]

Основные вызовы и ограничения Jenkins

- **Безопасность**: частые уязвимости требуют регулярного обновления и контроля (например, CVE-2025-5115 и другие). ^[148]
- **Сложная настройка масштабируемости и ресурсов** при большом количестве задач/агентов, особенно для современных Kubernetes-нативных задач.
- **Архаичность интерфейса**: устаревший UX/UI по сравнению с новыми решениями.
- **Высокий порог входа** для сложных pipeline и нестандартных сценариев.

- В эпоху cloud-native, GitOps и serverless многие выбирают альтернативы для новых проектов, хотя Jenkins по-прежнему незаменим в гибридных/традиционных средах. [\[144\]](#)

Jenkins — неизменный стандарт CI/CD, глубоко интегрирован в инфраструктуру большинства ИТ-корпораций, с динамичным развитием даже спустя 20 лет после появления. Но современные вызовы требуют всё большей адаптации к новым DevOps-тенденциям. [\[142\]](#) [\[148\]](#) [\[140\]](#) [\[144\]](#)



Оценить инструменты мониторинга и безопасности для Jenkins в 2025

В 2025 году инструменты мониторинга и безопасности Jenkins значительно развились, сосредоточившись на комплексной DevSecOps-защите, интеграции с системами наблюдаемости и автоматическом обнаружении уязвимостей. Jenkins остаётся ядром CI/CD-инфраструктур, поэтому контроль за его безопасностью и устойчивостью — критически важный аспект. [\[151\]](#) [\[152\]](#) [\[153\]](#) [\[154\]](#) [\[155\]](#)

Основные инструменты и подходы к мониторингу Jenkins

1. Prometheus + Grafana

- Дефакто-стандарт мониторинга Jenkins.
- С помощью плагина *Jenkins Prometheus Metrics Plugin* можно собирать ключевые метрики: длина очереди, загрузка агентов, процент успешных сборок, использование ресурсов JVM. [\[153\]](#) [\[151\]](#)
- Grafana визуализирует эти данные, обеспечивает SLA-дэшборды и алерты.
- Node Exporter добавляет мониторинг на уровне ОС — CPU, сетевые задержки, память.

2. Datadog, New Relic, Zabbix

- Используются предприятиями для многоуровневой аналитики CI/CD.
- Поддерживают машинное обучение для детектирования аномалий в загрузке агентов и задержках выполнения задач, с сокращением MTTR до 40%. [\[153\]](#)

3. Встроенные плагины Jenkins

- *Monitoring Plugin* — отображает статистику задач и ресурсов в интерфейсе Jenkins.
- *Support Core Plugin* — предоставляет отчёты о состоянии системы для аудита.
- *Audit Trail Plugin* — отслеживает API-вызовы, действия пользователей, что важно для соответствия требованиям compliance. [\[153\]](#)

4. Интеграции с SIEM и алертинг-системами

- Jenkins обычно подключают к Slack, PagerDuty, Telegram, email и SIEM-платформам (Splunk, Elastic Security) для автоматического реагирования на инциденты.

Инструменты и решения для безопасности Jenkins

1. Hygeni DevSecOps Suite

- Поддерживает Jenkins наряду с GitHub Actions и GitLab CI/CD.
- Обеспечивает SAST, SCA, IaC-сканирование, контроль контейнеров и автоматическое исправление уязвимостей.
- Генерирует SBOM (CycloneDX, SPDX) и блокирует рискованный код прямо на уровне pipeline-а.^[156]

2. OWASP Dependency-Check

- Проверяет зависимости плагинов и сборочных модулей на наличие уязвимостей (CVE).
- Может работать совместно с CI-этапами Jenkins Pipeline.

3. Anchore и Trivy

- Контейнерные сканеры, анализирующие образы Docker Jenkins и сборочные артефакты.
- Интегрируются в pipeline для гарантии безопасности контейнерных окружений.

4. Jenkins Security Advisories и безопасные плагины

- Сообщество регулярно публикует отчёты (в марте 2025 — обнаружены XSS, CSRF и хранения токенов в открытом виде в 25 плагинах).^[152]
- В августе 2025 года подтвержден критический эксплойт CVE-2025-53652, позволявший удалённое выполнение кода.^[155]
- Для противодействия компаниям рекомендуют ограничивать доступ к API, использовать Role-Based Access Control и обновлять плагины только из доверенных источников.

Обзор уровней защиты Jenkins в 2025

Уровень	Решения
Сеть и инфраструктура	Reverse-proxy (Nginx), VPN, Kubernetes Secrets, HashiCorp Vault
Контейнеры и образы	Trivy, Anchore, Clair
Код и зависимости	Hygeni, OWASP Dependency-Check
Pipeline и политика CI/CD	RBAC, secrets masking, подписанные агенты
Мониторинг и реакция	Prometheus, Grafana, Datadog, Audit Trail, SIEM-алерты

Основные вызовы Jenkins по безопасности в 2025

- **Повышенная уязвимость плагинов.** Более половины инцидентов безопасности связано с устаревшими или непроверенными плагинами. ^[152]
- **Масштабируемость мониторинга.** На больших кластерах (100+ агентов) классический Monitoring Plugin не справляется, нужны внешние системы Prometheus/Grafana. ^[153]
- **Человеческий фактор.** Отсутствие RBAC и неактуальные политики доступа остаются частыми причинами компрометации CI/CD.
- **Высокие требования к обновлениям.** Новые релизы часто содержат важные security-фиксы, но не все компании оперативно их внедряют.

Итог:

В 2025 году Jenkins остаётся мощным, но требующим системного подхода инструментом CI/CD. Для стабильной работы рекомендуется комбинировать **Prometheus + Grafana** для мониторинга и **Xygeni + OWASP Dependency-Check** для безопасности, а также применять **RBAC, SIEM и контейнерное сканирование** для комплексной DevSecOps-защиты. ^{[156] [151] [155] [152] [153]}



1. <https://blog.skillfactory.ru/vizualizatsiya-dannyh-v-python-s-pomoschyu-plotly/>
2. <https://practicum.yandex.ru/blog/plotly-dlya-vizualizacii-dannyh-v-python/>
3. <https://cartetika.ru/tpost/vt3drrdy71-grafiki-i-karti-vozmozhnosti-biblioteki>
4. <https://kedu.ru/press-center/articles/info-prog-kratkiy-spravochnik-po-sozdaniyu-grafikov-v-python-s-ispolzovaniem-plotly/>
5. [https://colab.research.google.com/github/dm-fedorov/pandas_basic/blob/master/быстрое введение в pandas/Взгляд на Plotly.ipynb](https://colab.research.google.com/github/dm-fedorov/pandas_basic/blob/master/быстрое%20введение%20в%20pandas/Взгляд%20на%20Plotly.ipynb)
6. <https://external.software/archives/18540>
7. <https://habr.com/ru/articles/502958/>
8. <https://python-school.ru/blog/визуализация-данных/plotly-basics/>
9. [https://dfedorov.spb.ru/pandas/Взгляд на Plotly.html](https://dfedorov.spb.ru/pandas/Взгляд_на_Plotly.html)
10. <https://habr.com/ru/companies/skillfactory/articles/506974/>
11. <https://blog.skillfactory.ru/vizualizatsiya-dannyh-v-python-s-pomoschyu-plotly/>
12. <https://practicum.yandex.ru/blog/plotly-dlya-vizualizacii-dannyh-v-python/>
13. <https://cartetika.ru/tpost/vt3drrdy71-grafiki-i-karti-vozmozhnosti-biblioteki>
14. <https://habr.com/ru/companies/garage8/articles/908950/>
15. <https://external.software/archives/18540>
16. <https://eddydewrussia.ru/2022/05/25/22/52/5289/>
17. <https://habr.com/ru/companies/skillfactory/articles/506974/>
18. <https://course.rintro.ru/plotly.html>
19. <https://soware.ru/products/plotly-dash>

20. <https://habr.com/ru/articles/502958/>
21. <https://blog.skillfactory.ru/vizualizatsiya-dannyh-v-python-s-pomoschyu-plotly/>
22. <https://practicum.yandex.ru/blog/plotly-dlya-vizualizatsii-dannyh-v-python/>
23. <https://kedu.ru/press-center/articles/info-prog-kratkiy-spravochnik-po-sozdaniyu-grafikov-v-python-s-ispolzovaniem-plotly/>
24. <https://habr.com/ru/articles/752406/>
25. <https://habr.com/ru/companies/otus/articles/716190/>
26. <https://habr.com/ru/companies/skillfactory/articles/510320/>
27. <https://habr.com/ru/articles/456282/>
28. <https://learn.microsoft.com/ru-ru/kusto/functions-library/plotly-scatter3d-fl?view=microsoft-fabric>
29. <https://sky.pro/wiki/analytics/sozdanie-interaktivnyh-tochechnykh-diagramm-s-plotly-obzor-scatter/>
30. <https://habr.com/ru/articles/502958/>
31. <https://soware.ru/products/plotly-dash>
32. <https://habr.com/ru/companies/otus/articles/716190/>
33. <https://itproger.com/course/data-analytics/6>
34. <https://is42-2018.susu.ru/poselenovia/2020/03/20/kratkoe-rukovodstvo-po-dash-python-veb-frejmvo-rk-dlya-sozdaniya-deshbordov/>
35. <https://victor-komlev.ru/frejmworki-plotly-i-dash/>
36. <https://open.zeba.academy/sozdaniye-interaktivnykh-paneley-dannymi-pomoshyu-plotly-dash/>
37. <https://habr.com/ru/articles/431754/>
38. <https://leftjoin.ru/blog/bi-tools/dash-bi-guide/>
39. <https://blog.skillfactory.ru/vizualizatsiya-dannyh-v-python-s-pomoschyu-plotly/>
40. <https://soware.ru/compare/plotly-dash-vs-nexign-request>
41. <https://is42-2018.susu.ru/poselenovia/2020/03/20/kratkoe-rukovodstvo-po-dash-python-veb-frejmvo-rk-dlya-sozdaniya-deshbordov/>
42. <https://proglab.io/p/tutorial-vizualizatsiya-dannyh-v-vebe-s-pomoshchyu-python-i-dash-2021-01-11>
43. <https://habr.com/ru/articles/663686/>
44. <https://itproger.com/course/data-analytics/6>
45. <https://www.netangels.ru/support/cloud-hosting/dash/>
46. <https://kedu.ru/press-center/articles/info-prog-sozdavayte-interaktivnye-vizualizatsii-dannykh-s-pomoshchyu-python-dash/>
47. <https://habr.com/ru/companies/otus/articles/716190/>
48. <https://habr.com/ru/articles/868766/>
49. <https://habr.com/ru/articles/47474/>
50. https://www.reddit.com/r/learnprogramming/comments/199s8db/the_difference_between_dash_and_python/
51. <https://practicum.yandex.ru/blog/plotly-dlya-vizualizatsii-dannyh-v-python/>
52. <https://python-school.ru/blog/визуализация-данных/plotly-basics/>
53. <https://itproger.com/course/data-analytics/6>
54. <https://habr.com/ru/articles/502958/>

55. <https://habr.com/ru/companies/skillfactory/articles/506974/>
56. <https://habr.com/ru/companies/ods/articles/323210/>
57. <https://bigdataschool.ru/blog/mlops-and-mlflow-best-practices/>
58. https://kz.hexlet.io/lesson_filters/plotly
59. https://www.reddit.com/r/MachineLearning/comments/r8ob3o/p_visualize_millions_of_datapoints_using_plotly/
60. <https://habr.com/ru/companies/oleg-bunin/articles/455648/>
61. <https://habr.com/ru/companies/skillfactory/articles/710816/>
62. <https://habr.com/ru/articles/900788/>
63. <https://bigdataschool.ru/blog/mlops-and-mlflow-best-practices/>
64. <https://habr.com/ru/companies/otus/articles/883894/>
65. <https://kolodezev.ru/mlsysd11.html>
66. <https://habr.com/ru/articles/588645/>
67. <https://codenrock.com/blog/vybiraem-stek-dlya-ml-sorevnovaniya-moshhnye-resheniya-na-baze-otkrytyh-tehnologii/>
68. https://www.reddit.com/r/MachineLearning/comments/1nzw0v3/d_best_practices_for_structuring_an_applied_ml/
69. <https://edgcenter.ru/blog/infrastructura-dlya-ml-proectov>
70. <https://intuit.ru/studies/courses/48/48/lecture/1456>
71. <https://bigdataschool.ru/blog/ml-pipelines-orchestration-with-airflow/>
72. <https://habr.com/ru/articles/737046/>
73. <https://habr.com/ru/companies/vk/articles/720654/>
74. <https://habr.com/ru/articles/749178/>
75. <https://habr.com/ru/companies/otus/articles/883894/>
76. <https://habr.com/ru/companies/wunderfund/articles/836908/>
77. <https://habr.com/ru/articles/692462/>
78. <https://www.glukhov.org/ru/post/2025/06/apache-airflow/>
79. <https://bigdataschool.ru/blog/airflow-and-ray-for-mlops/>
80. <https://www.youtube.com/watch?v=NfPf0Y770DA>
81. <https://blog.skillfactory.ru/glossary/apache-airflow/>
82. <https://kurshub.ru/journal/blog/chto-takoe-apache-airflow/>
83. <https://bigdataschool.ru/wiki/airflow/>
84. <https://bigdataschool.ru/blog/news/airflow/go-sdk-in-airflow-3/>
85. <https://www.securitylab.ru/blog/personal/SimlpeHacker/355517.php>
86. https://www.reddit.com/r/dataengineering/comments/x8ufv3/is_there_anyway_to_configure_apache_airflow/
87. https://ya.ru/neurum/c/tehnologii/q/kak_apache_airflow_otlichaetsya_ot_drugih_instrumentov_198c1366
88. <https://habr.com/ru/companies/vk/articles/339392/>
89. <https://practicum.yandex.ru/blog/apache-airflow/>

90. <https://habr.com/ru/companies/alfa/articles/676926/>
91. <https://hevodata.com/learn/airflow-vs-jenkins/>
92. https://www.reddit.com/r/dataengineering/comments/oh5vq0/going_from_jenkins_to_airflow/
93. <https://kurshub.ru/journal/blog/chto-takoe-apache-airflow/>
94. <https://learn.microsoft.com/ru-ru/azure/databricks/machine-learning/mlops/ci-cd-for-ml>
95. https://www.reddit.com/r/dataengineering/comments/rft5k9/airflow_vs_jenkins/
96. <https://bigdataschool.ru/blog/ci-cd-dag-airflow-approaches-and-cases/>
97. <https://habr.com/ru/companies/alfa/articles/821447/>
98. <https://bigdataschool.ru/blog/airflow-vs-vertex-ai-pipelines-in-mlops/>
99. <https://habr.com/ru/companies/alfa/articles/947754/>
100. https://ya.ru/neurum/c/tehnologii/q/kak_apache_airflow_otlichaetsya_ot_drugih_instrumentov_198c1366
101. <https://bigdataschool.ru/blog/airflow-vs-luigi/>
102. <https://hevodata.com/learn/luigi-vs-airflow-key-comparison/>
103. <https://bigdataschool.ru/blog/airflow-vs-vertex-ai-pipelines-in-mlops/>
104. https://www.reddit.com/r/dataengineering/comments/rft5k9/airflow_vs_jenkins/
105. <https://kurshub.ru/journal/blog/chto-takoe-apache-airflow/>
106. <https://kgeu.ru/upload/iblock/e61/y3rm0l4be1wa536fmr0a59ye3fevsrm4/UMM.-Avtomatizatsiya-mashinnogo-obucheniya.pdf>
107. <https://glukhov.org/ru/post/2024/11/selfhosted-integration/>
108. https://www.reddit.com/r/dataengineering/comments/v4j863/opinion_airflow_luigi_or_both/
109. <https://otus.ru/lessons/python-professional/program/>
110. https://www.reddit.com/r/dataengineering/comments/rft5k9/airflow_vs_jenkins/
111. <https://www.projectpro.io/compare/apache-airflow-vs-luigi>
112. <https://hevodata.com/learn/airflow-vs-jenkins/>
113. <https://www.linkedin.com/pulse/2024-data-engineering-workflow-orchestration-tool-parameswaran-ltg6c>
114. <https://bigdataschool.ru/blog/airflow-vs-luigi/>
115. <https://bigdataschool.ru/blog/airflow-vs-vertex-ai-pipelines-in-mlops/>
116. <https://kurshub.ru/journal/blog/chto-takoe-apache-airflow/>
117. https://www.reddit.com/r/MachineLearning/comments/mgzvt2/d_whats_the_simplest_most_lightweight_but/
118. <https://www.keboola.com/blog/best-airflow-alternatives>
119. https://www.reddit.com/r/dataengineering/comments/rft5k9/airflow_vs_jenkins/
120. https://www.reddit.com/r/dataengineering/comments/n1qs11/what_are_the_best_frameworks_data_engineers/
121. <https://external.software/archives/20137>
122. <https://hevodata.com/learn/airflow-vs-jenkins/>
123. https://habr.com/ru/companies/lemana_tech/articles/580944/
124. <https://bigdataschool.ru/blog/airflow-vs-luigi/>

125. <https://habr.com/ru/companies/slurm/articles/708420/>
126. <https://bigdataschool.ru/blog/ci-cd-dag-airflow-approaches-and-cases/>
127. <https://kurshub.ru/journal/blog/chto-takoe-etl/>
128. https://www.reddit.com/r/dataengineering/comments/168p757/will_airflow_become_obsolete_in_coming_years/
129. <https://bigdataschool.ru/blog/airflow-vs-vertex-ai-pipelines-in-mlops/>
130. [https://ru.wikipedia.org/wiki/Jenkins_\(программное_обеспечение\)](https://ru.wikipedia.org/wiki/Jenkins_(программное_обеспечение))
131. <https://slurm.io/blog/jenkins-chto-eto-takoe-i-zachem-on-nuzhen>
132. <https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-jenkins/>
133. <https://bigdataschool.ru/blog/airflow-vs-luigi/>
134. https://ru.wikipedia.org/wiki/Apache_Airflow
135. <https://habr.com/ru/companies/lamoda/articles/518620/>
136. <https://habr.com/ru/companies/jugru/articles/441282/>
137. <https://javaswag.github.io/episode/45/>
138. <https://ru.ruwiki.ru/wiki/Jenkins>
139. https://ru.wikipedia.org/wiki/Манчинелли,_Луиджи
140. [https://ru.wikipedia.org/wiki/Jenkins_\(программное_обеспечение\)](https://ru.wikipedia.org/wiki/Jenkins_(программное_обеспечение))
141. <https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-jenkins/>
142. <https://slurm.io/blog/jenkins-chto-eto-takoe-i-zachem-on-nuzhen>
143. <https://blog.skillfactory.ru/dzhenkins-chto-eto-za-programma-i-dlya-chego-ona-nuzhna/>
144. <https://dstglobal.ru/club/1117-jenkins-v-epohu-kubernetes>
145. <https://kurshub.ru/journal/blog/chto-takoe-apache-airflow/>
146. <https://bigdataschool.ru/blog/airflow-vs-luigi/>
147. <https://community.jenkins.io/t/governance-meeting-january-13-2025/25706>
148. https://1275.ru/vulnerability/kriticheskie-uyazvimosti-v-jenkins-eksperty-rekomenduyut-srochnoe-obnovlenie_15058
149. <https://habr.com/ru/companies/jugru/articles/441282/>
150. <https://www.youtube.com/watch?v=pgCPYVSioKY>
151. <https://habr.com/ru/companies/slurm/articles/788836/>
152. <https://www.itsec.ru/news/komanda-jenkins-preduprdila-ob-uyazvimost-v-25-plaginah>
153. <https://moldstud.com/articles/p-essential-jenkins-configuration-metrics-you-should-monitor-for-optimal-performance>
154. <https://habr.com/ru/companies/otus/articles/831962/>
155. https://1275.ru/vulnerability/kriticheskaya-uyazvimost-v-jenkins-1-500-serverov-pod-ugrozoy-udalennogo-vypolneniya-koda_13975
156. <https://xygeni.io/ru/blog/top-devops-security-tools/>
157. <https://www.browserstack.com/guide/jenkins-best-practices-every-developer-must-know>
158. <https://core247.kz/blog/top-16-devops-tools-for-2025>
159. <https://vc.ru/services/2288281-instrumenty-devsecops-dlya-bezopasnosti-v-2025-godu>

160. <https://uzcert.uz/ru/10-devops-2025/>