

# Лекция 3

# System Design и Agile

Проектирование систем и продуктовая веб-разработка

Канев Антон Игоревич

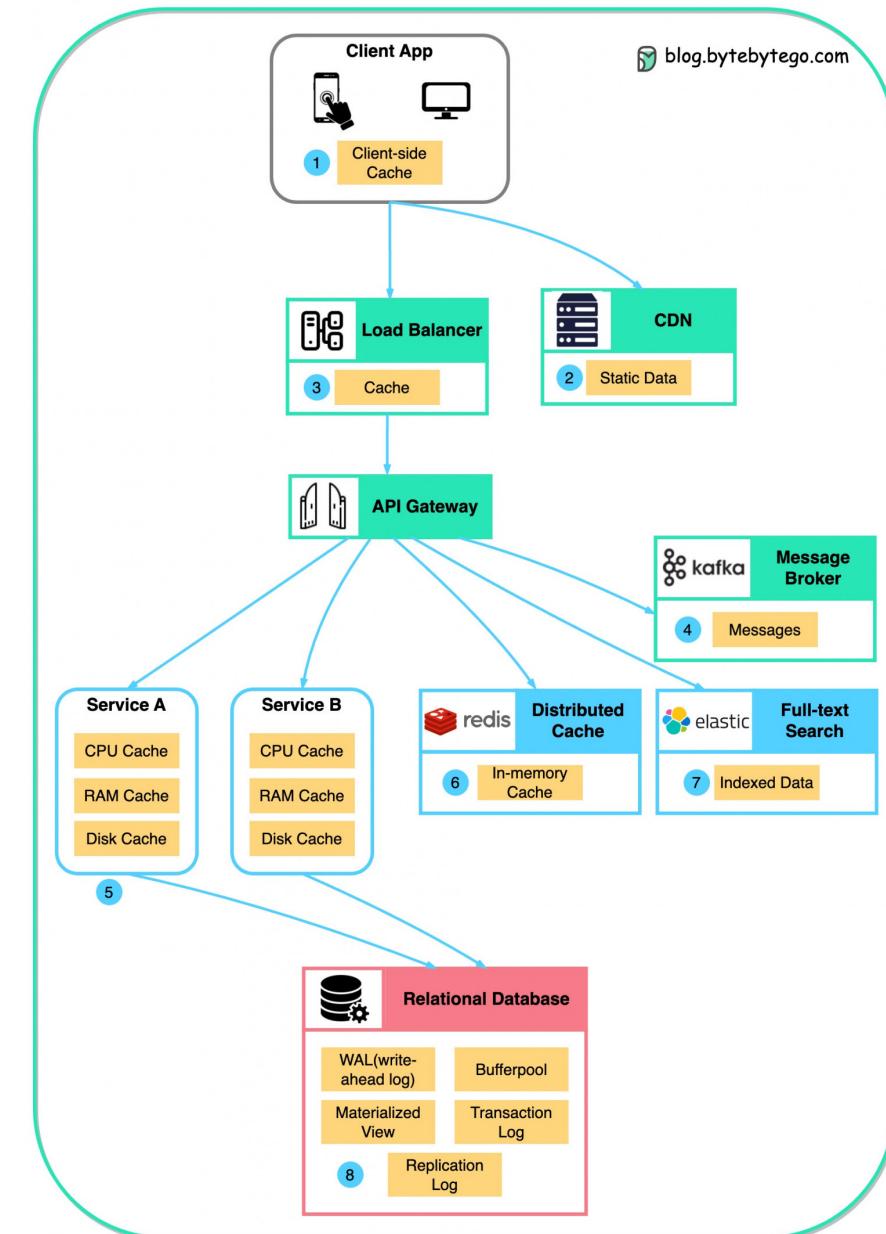
# Дизайн системы

## или зачем нужны диаграммы

- Архитектура больших систем очень сложна
- Очень хорошая шпаргалка приведена по ссылке ниже
- В курсе мы затронем часть аспектов, но чтобы понять их все требуется время и опыт

<https://habr.com/ru/articles/770564/>

Cache Systems Every Developer Should Know



## \*Зачем нужна документация

- Обучение новых сотрудников
- Снижение рисков при разработке продукта (System Design)
- Универсальный язык, слабо зависит от языка программирования
- Описание сложных системы и взаимодействия их между собой
- Поддержка продукта и его развертывание
- Low-code разработка

## \*Проект, продукт и процесс

- **Проект** – временное предприятие, направление на получение результата
- **Процесс** – повторяемая последовательность действий, направленная на достижение цели
- **Продукт** – программный продукт, программа
- Minimum viable product (**MVP**) – для гостя 1/5 лабораторная
  
- Цель – конечное желание (повысить продажи)
- Задачи – совокупность шагов для достижения цели

# \*Роли в команде

- Заказчик (знания, как автор книги)
  - Директор проекта (деньги, инвестор)
  - Руководитель проекта
  - UI/UX-дизайнер
  - Бизнес аналитик
  - Системный аналитик
  - Руководитель команды
  - Разработчик
  - Тестировщик QA
  - DevOps/SRE
- Бизнес-Аналитик (как сценарист)**
- ближе к заказчику
  - описание функций системы
  - описание пользователей и их ролей
  - интерфейс системы
- Макет (если система для внутреннего пользования)
- Артефакты
  - Описание бизнес процессов
  - BPMN или Activity
  - Use-case

# \*Менеджер проекта

Engineering / Roadmap

Share Updates Favorite ...

По сути это продюсер – следит за **сроками и ищет/урезает деньги**

- Понимание конечной цели - в больших проектах часто все против вас: сотрудники заказчика (бизнес), другие команды разработчиков
- Протоколы совещаний
- Контроль сроков: вовремя утвердить ТЗ, макет figma, вовремя показать MVP и тд
- Дорожная карта (Roadmap)
- Kanban-доска

## Roadmap

By Status ▾

Properties Group by Status Filter Sort Search ... New ▾

Not Started 12

>Login Bugs  
Camille Ricketts  
Nate Martins  
Bug 🐞  
Sprint 20

In Progress 5

Onboarding  
Andrea Lim  
Nate Martins  
Epic 🚀

Complete 5

Rewrite Query Caching Logic  
David Tibbitts  
Cory Etzkorn  
Task ↗  
Sprint 23 Sprint 24

Hidden

No

Mobile Start Up Time

Camille Ricketts  
David Tibbitts Andrea Lim  
Epic 🚀

Error Codes

Andrea Lim Cory Etzkorn  
Bug 🐞  
Sprint 21

Rewriting Flow

David Tibbitts  
Camille Ricketts Nate Martins  
Andrea Lim  
Task ↗  
Sprint 23

Mobile

Ben Lang  
Cory Etzkorn  
Task ↗  
Sprint 23

Landing Page Redesign

Cory Etzkorn  
Nate Martins Andrea Lim

Sharing Permissions

David Tibbitts  
Cory Etzkorn

- **Kanban** - способ организации множества задач: То что нужно сделать описано в user stories (я-хочу-результат)
- Notion, Trello, Taiga, Jira и тд (**task managers**)

# \*Дорожная карта

Это более высокий уровень

- Пользовательские истории группируются в **эпики** – конкретные результаты : курс по РИП или диплом
- В эпике может участвовать несколько команд
- В конечном итоге нас интересует **тема** (цель) – оцифровать кампус университета
- Для ее достижения формируются **инициативы**



# \*Руководитель команды и разработчик

## **Team lead (агент для наших актеров)**

- Набор и развитие команды
- Административные вопросы
- Отдельный вид – **технический руководитель** (архитектура)
- Развитие платформы, архитектура системы

## **Разработчик (как актер)**

- Frontend – Java Script, TS, React, Vue.js
- Backend – Java, C#, Go, Python и тд. SQL
- Дата инженер – создание хранилища данных, SQL+ETL
- Дата аналитик – выявление зависимостей в данных, построение графиков и тд
- Data Scientist – специалист по ML моделям

# DevOps и Тестировщик

- QA – ручное, но чаще автоматическое через Python
- Альфа-тестирование
- Не путать с unit-тестированием и автотестами, его пишет сам разработчик
- интеграционное тестирование
- нагрузочное тестирование
- Devops – процесс переноса кода с компьютера разработчика до конечного пользователя
- это как процесс монтажа и получения готового файла для фильма
- SRE – админы серверов и виртуальных машин
- Выделяют под наши просьбы виртуальные машины, назначают им адреса
- Заказывают новое оборудование в датацентр

# \*Системный аналитик

- Это профессия нашей кафедры
- Как режиссер, от него зависит, что напишут разработчики
- ближе к разработчикам
- описание данных
- описание алгоритмов
- архитектура системы
- ER диаграмма и описание БД
- UML диаграммы: развертывание, последовательности
- Описание списка запросов: таблицей или swagger



# Техническое задание

## Функциональные требования

- Описание элементов интерфейса и доступных действий для разных ролей пользователей
- Группировка по страницам интерфейса
- При описании интерфейса либо лучше сослаться на методы, либо в методах сослаться на конкретные пункты функций

– Неавторизованному пользователю доступен только список акций.

5.1.1.4 Разработка страницы просмотра котировок и графика цены акций конкретной компании:

- На странице пользователь получает данные о компании: текущую цену акции, график и описание компании;
- Авторизованному пользователю также доступны функции покупки и продажи.

5.1.1.5 Разработка страницы добавления и редактирования компаний:

- Доступ к странице имеют только менеджеры;
- На странице можно добавлять, удалять и редактировать имеющиеся компании.

# ТЗ

- Техническое задание описывает все аспекты реализации системы
- Фиксирует функционал системы на понятном заказчику и разработчикам языке
- Включает ряд приложений: описание БД, списка методов (из swagger), figma

## Таблица атрибутов пользователя (surdoapi\_userattrs)

Используется для добавления атрибутов с сохранением стандартной модели пользователя

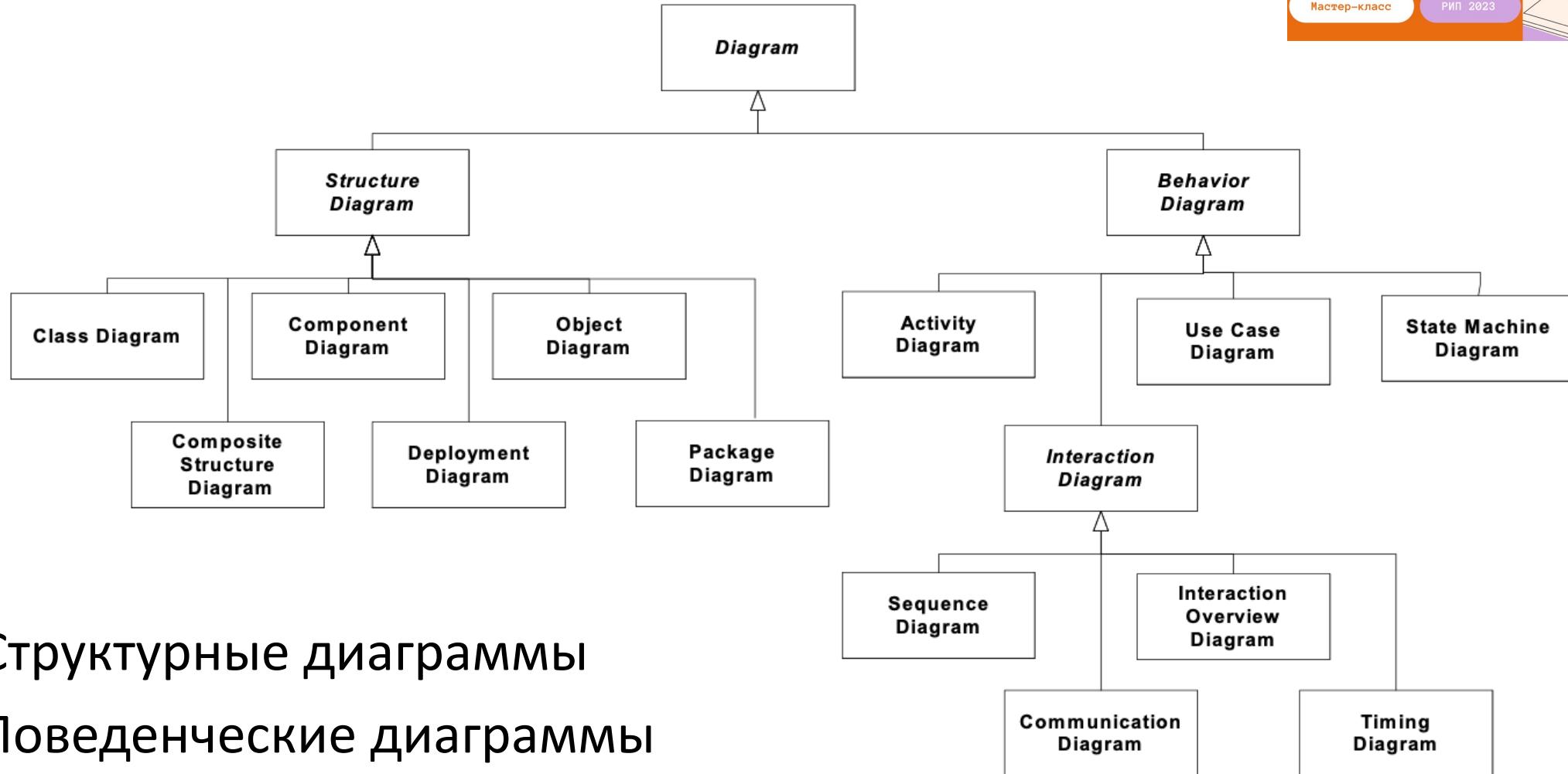
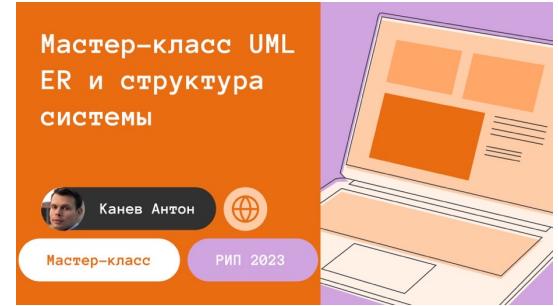
| Столбец  | Тип           | Модификаторы | Описание                              |
|----------|---------------|--------------|---------------------------------------|
| id       | BIGINT        | PK           | id атрибута                           |
| user_id  | BIGINT        | FK           | id пользователя                       |
| key      | VARCHAR       |              | Название атрибута                     |
| value    | VARCHAR       |              | Значение атрибута                     |
| end_date | TIMESTAMP (6) |              | Дата, до которой действительна запись |

# Т3. Описание методов

- Описание методов таблицей. Также описание можно подготовить в виде swagger
- Методы группируются по их домену url: «/user/..», «/lesson/..», «/request/..»
- Выходные данные часто аналогичны GET, чтобы сразу отобразить данные в интерфейсе

| Название | Описание                            | Method | Path                         | Вход   | Выход  |
|----------|-------------------------------------|--------|------------------------------|--|--|
|          | Добавление нового задания в занятие | POST   | /lesson/<int:lesson_id>/task | {<br>id: integer<br>task_text: string,<br>task_audio: string <URL><br>task_type: string<br>} | [<br>{<br>id: integer<br>task_text: string,<br>task_audio: string <URL><br>task_type: string<br>}<br>] |

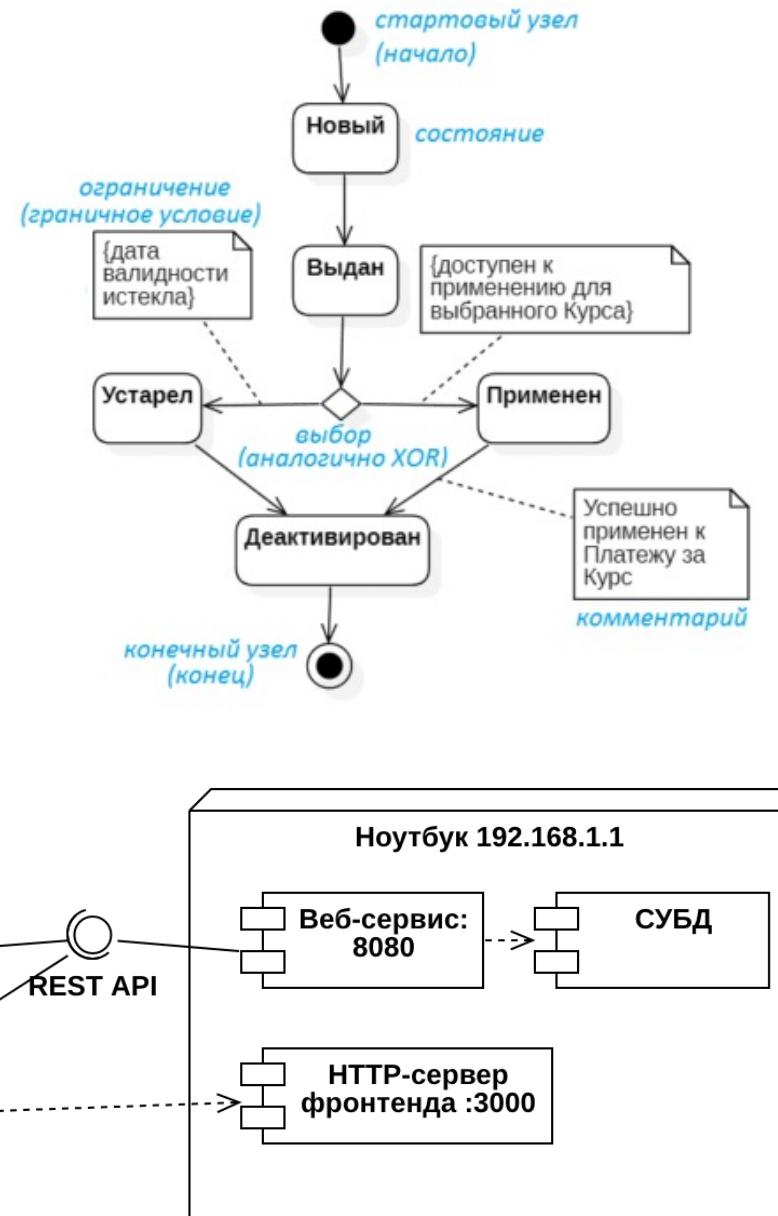
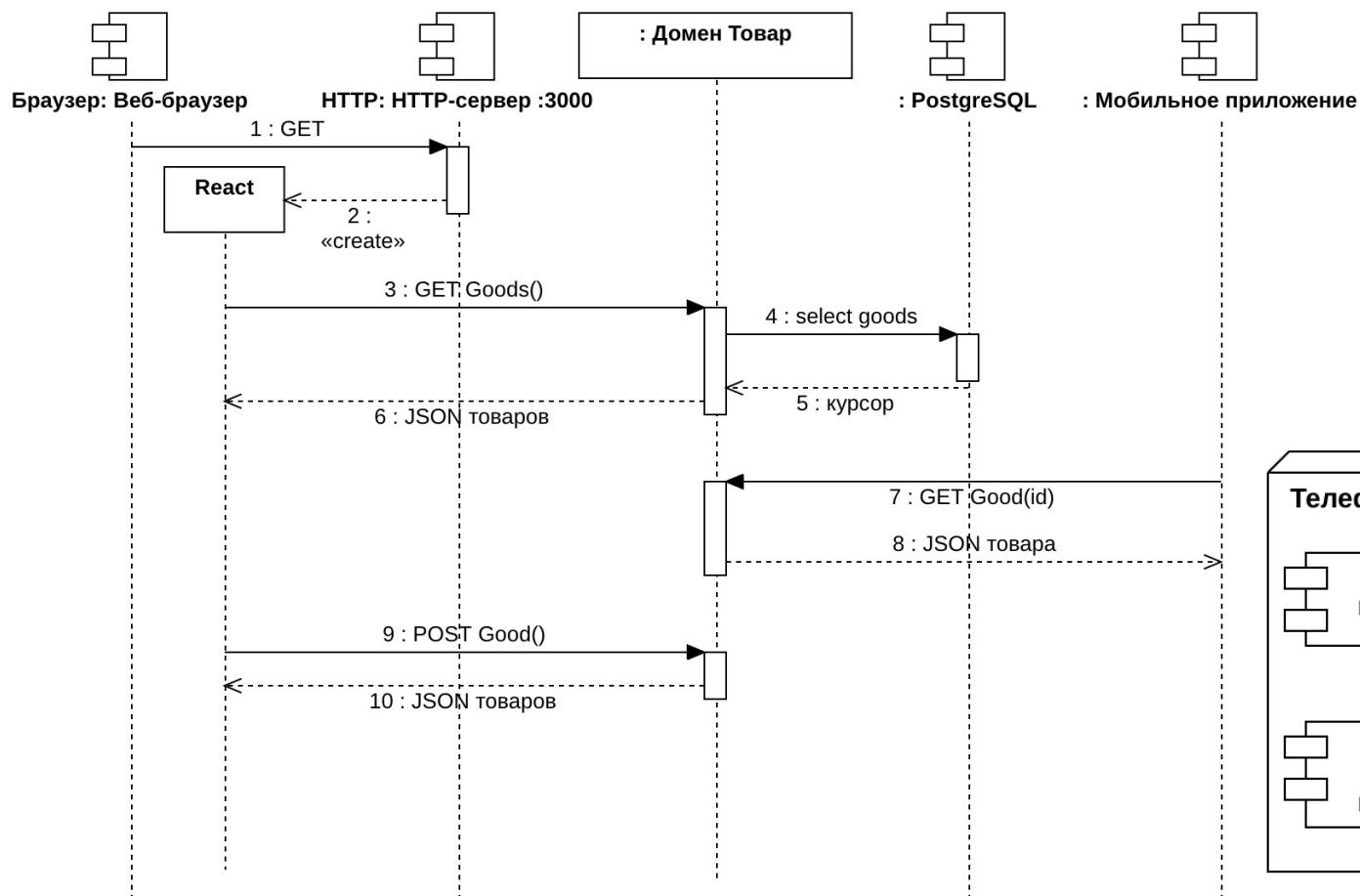
# \* Виды диаграмм UML



- Структурные диаграммы
- Поведенческие диаграммы

# Sequence, Deployment, Activity

- В StarUML своя MVC: вы добавляете класс на одну диаграмму, а его методы можно вызвать в другой



# \*Agile – давайте снимать сериал

- При фиксации двух параметров получаем третий
- Достичь сразу всех трех не получится, нужно перейти на **новый уровень**

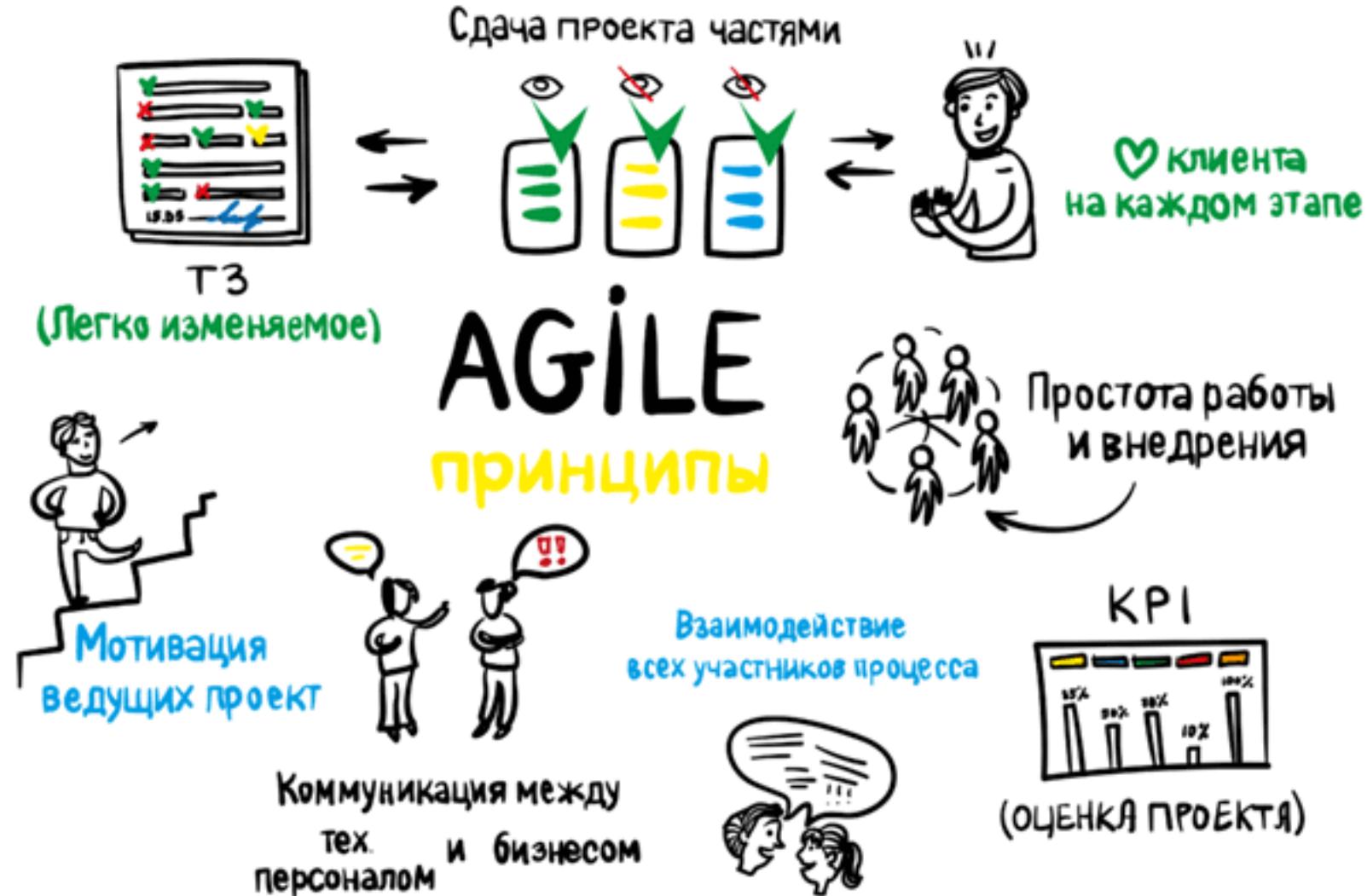


- Гибкая методология разработки – альтернатива последовательной водопадной
- Разделение процесса разработки на короткие итерации и повторение



# Agile

- Гибкая методология разработки – альтернатива последовательной водопадной
- Разделение процесса разработки на короткие итерации и повторение



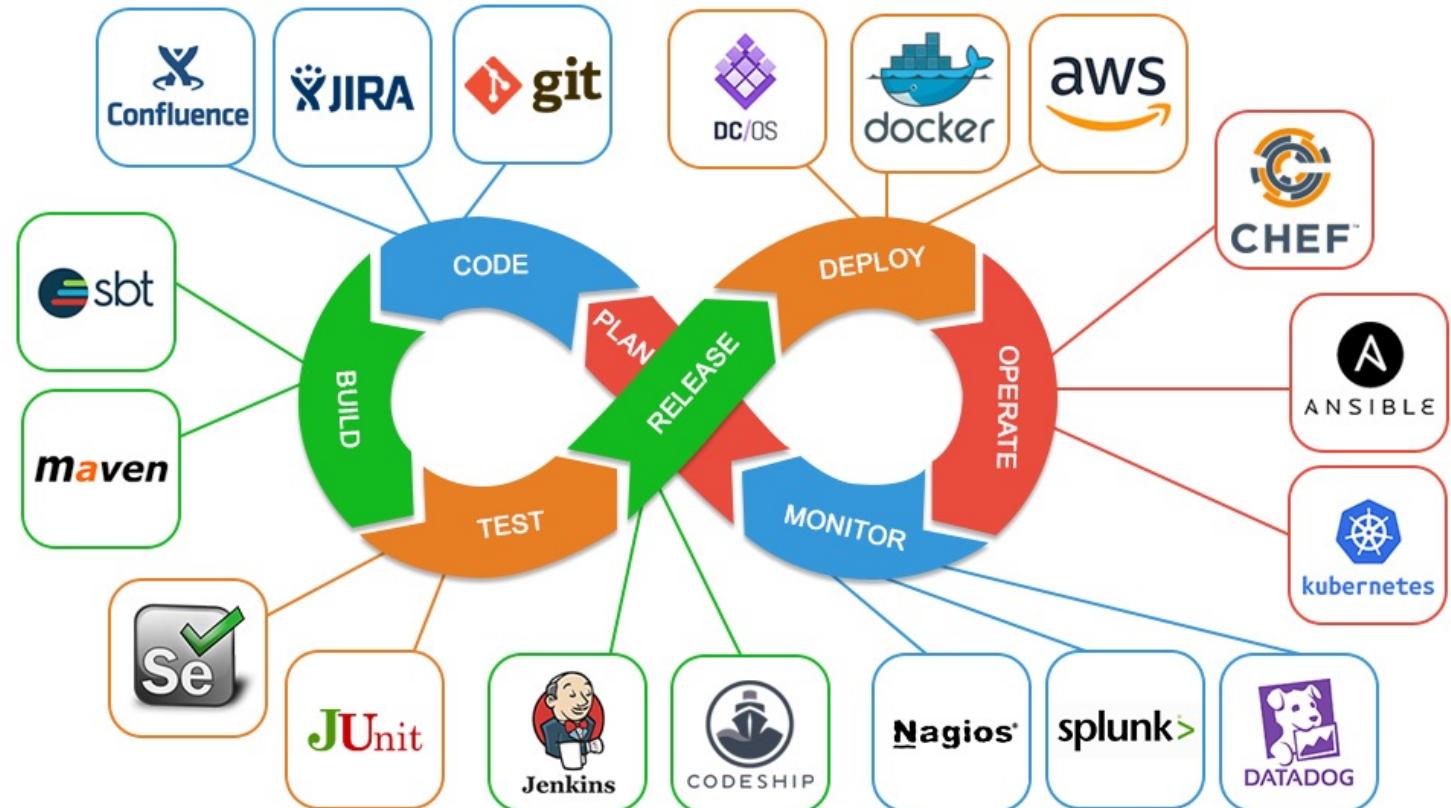
# Scrum и Бэклог

- Вообще **Scrum** о том как работать без руководителя: scrum-мастер и владелец продукта только выполняют роли/правила, но не руководят
- Нас интересует **бэклог**
- Каждые 2 недели формируем задачи на спринт



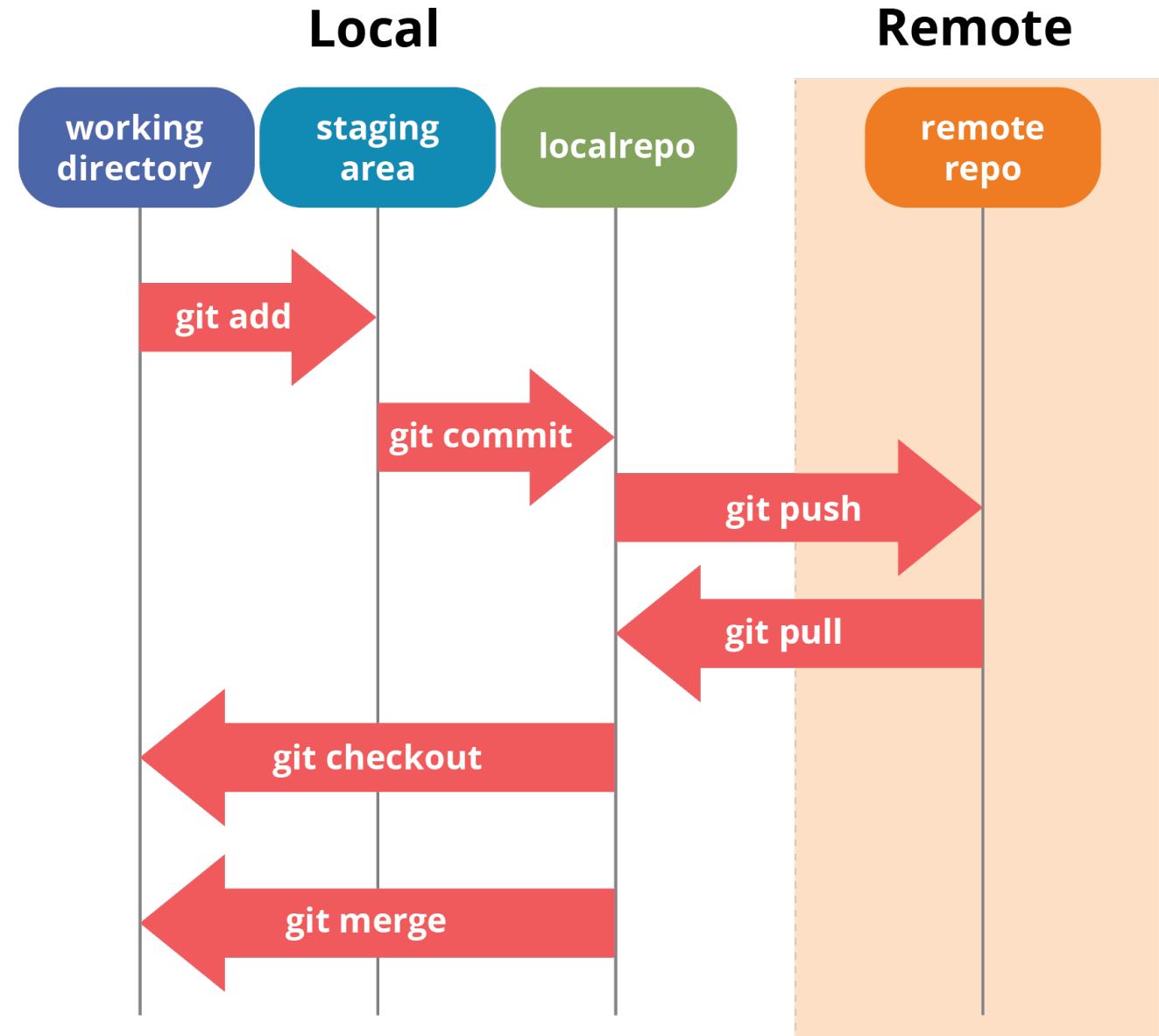
# \*DevOps – воплощение Agile на практике

- DevOps (development & operations) — методология автоматизации технологических процессов сборки, настройки и развёртывания программного обеспечения
- Быстрый перенос программного обеспечения между разными стадиями жизненного цикла ПО
- Снижение частоты отказов
- Сокращение времени доработок



# \*Git

- Git – распределенная система управления версиями
- Позволяет хранить несколько версий одного и того же документа

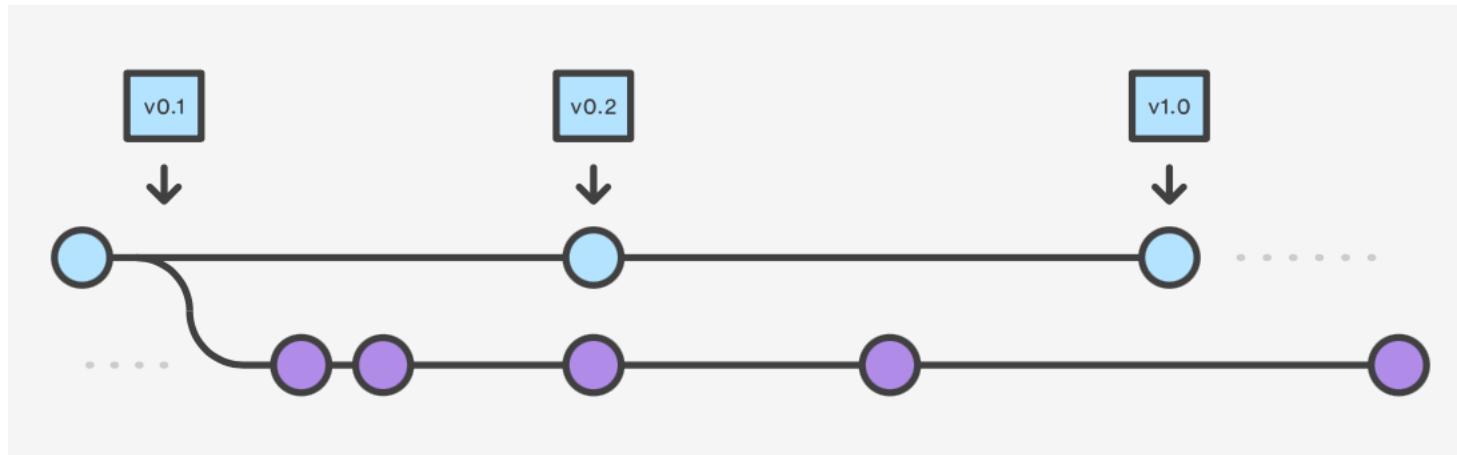


# Develop

```
git branch develop
```

```
git push -u origin develop
```

- У нашей разработки есть версии
- До v1.0 бета-тестирование
- Main/master – это версия продукта у пользователей



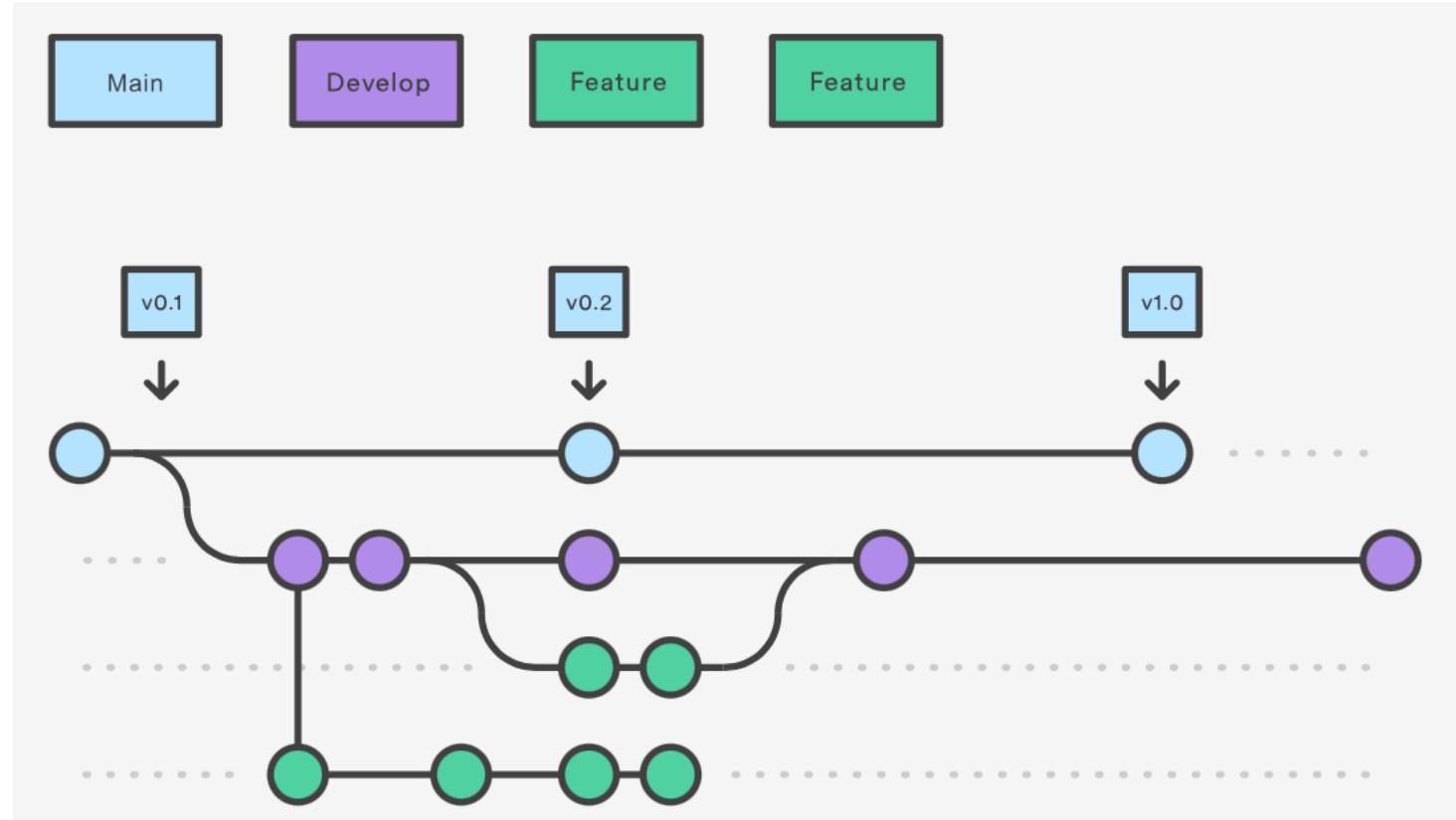
# Feature

```
git checkout develop
```

```
git checkout -b feature_branch
```

```
git checkout develop
```

```
git merge feature_branch
```



- Develop – единая стабильная версия для разработчиков
- Feature – много веток, каждая со своими новыми функциями

# Release

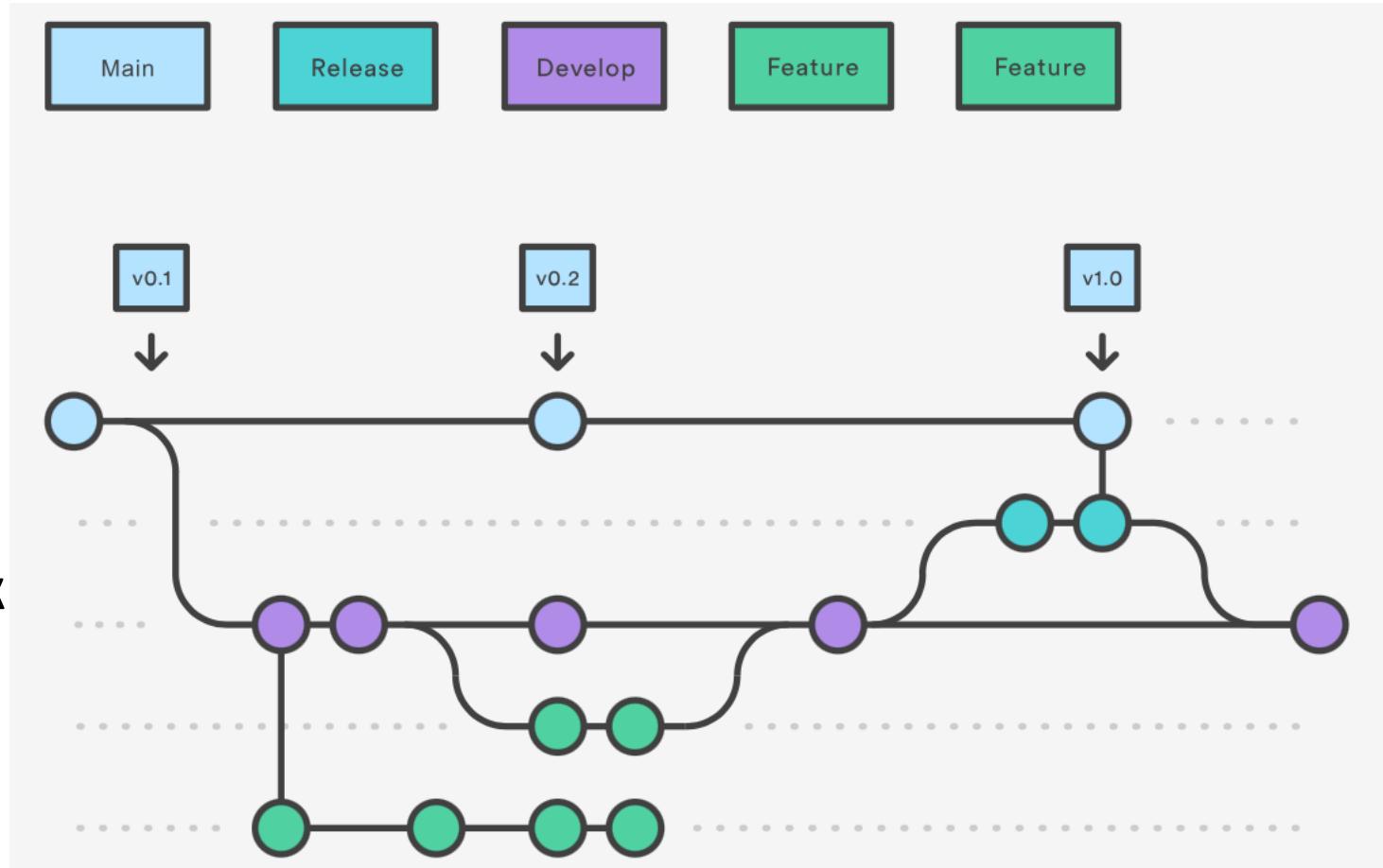
```
git checkout develop
```

```
git checkout -b release/0.1.0
```

```
git checkout main
```

```
git merge release/0.1.0
```

- Когда несколько доработок из Feature готовы, то в конце спринта формируем **Release**
- У нас будет для GitHub Pages



# \*Hotfix

- Hotfix – критичные исправления

```
git checkout main
```

```
git checkout -b hotfix_branch
```

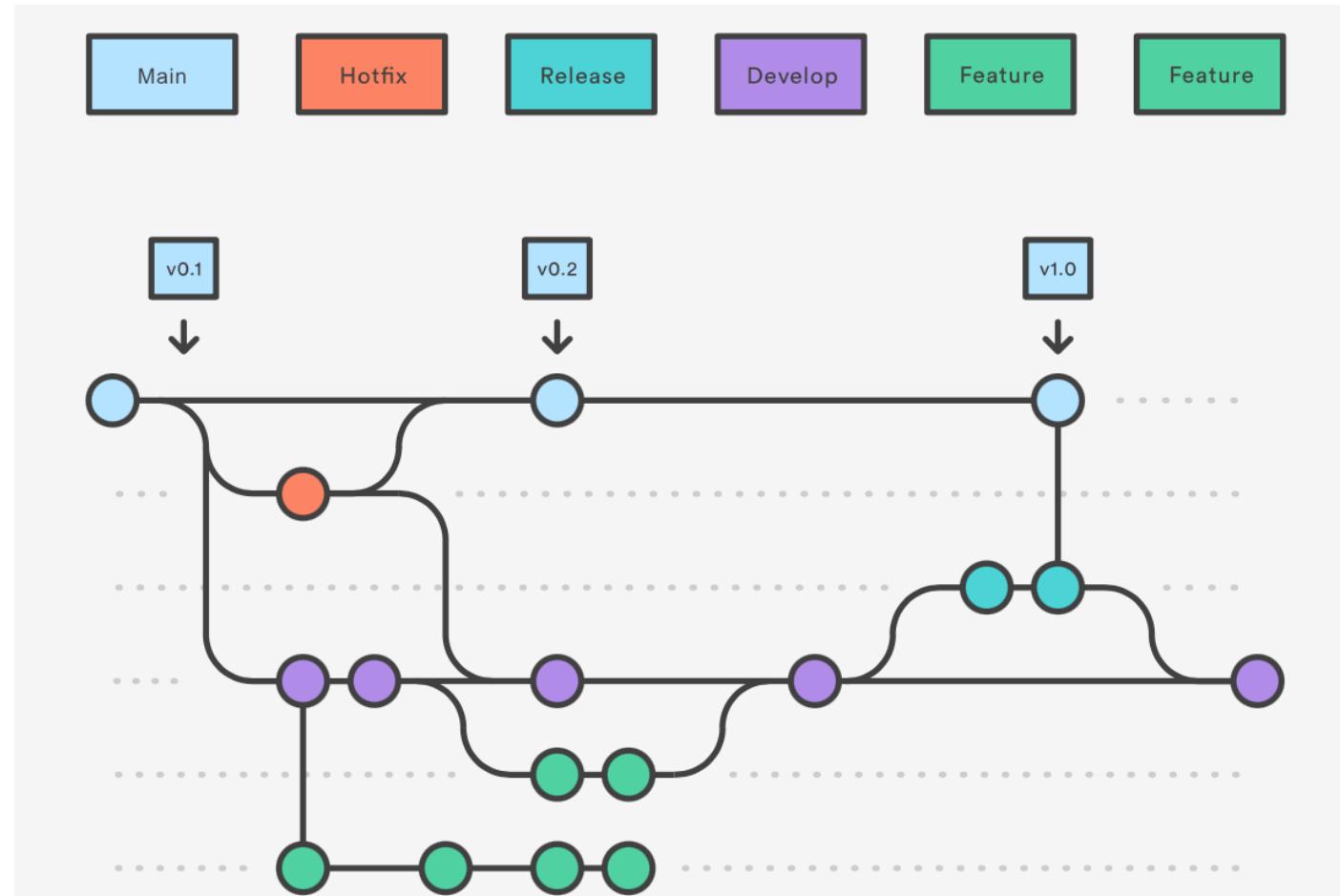
```
git checkout main
```

```
git merge hotfix_branch
```

```
git checkout develop
```

```
git merge hotfix_branch
```

```
git branch -D hotfix_branch
```



- Обязательная схема по Gitflow