# **Модели веток. Простая feature branch модель**

В предыдущих уроках мы объяснили, что не так с самым простым подходом с одной общей веткой main. Теперь разберём популярные подходы, которые отвечают задачам командной разработки и обычно не вызывают проблем.

Начнём с определения. **Подходы** **к работе с ветками** — это правила, которые описывают, когда и для чего создаются ветки, какие в них коммиты и в какой момент происходит слияние веток.

Допустим, вы разработчик и приходите в команду, где уже используют один из подходов. Чем быстрее вы поймёте, по каким правилам действуют ваши коллеги, тем проще вам будет освоиться на новом месте и влиться в процесс.

### **Подходы**

Рассмотрим три самых популярных подхода — их также принято называть **workflow** (англ. «рабочий процесс») или сокращённо: **flow**.

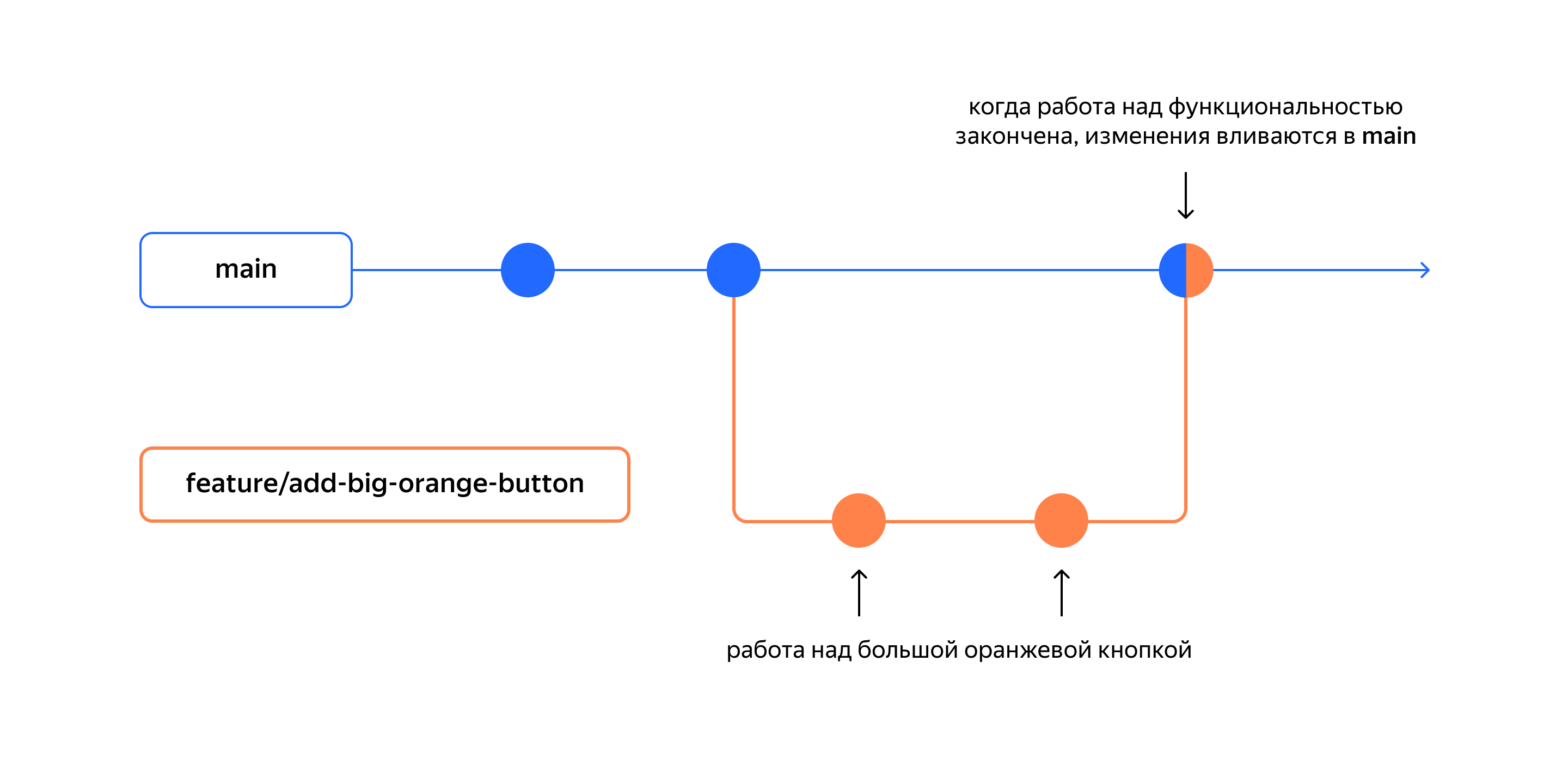
* **Feature branch workflow** — простой и самый популярный вариант. Если коротко, в нём для каждого нового изменения создаётся новая ветка, которая позже вливается в main с помощью git merge.
* **Git flow** — более сложный вариант. Подход похож на feature branch workflow, но в нём создаётся больше веток, а изменения (коммиты) делят на разные типы: исправление, новая функциональность и так далее. Разные типы коммитов попадают в разные ветки.
* **Trunk-based** — популярный в больших компаниях (таких как Яндекс, Google и прочих) подход, который обещает бо́льшую скорость работы в крупных командах. Этот подход тоже похож на feature branch workflow. Главное отличие в том, что участники проекта вливают (merge) свой код в основную ветку максимально часто. Например, каждый день.

Пока сосредоточимся на более простой модели — feature branch workflow. Важно понимать, что во всех подходах есть много общего. Если вы поймёте, как работает один из них, освоить другие будет несложно.

### **Feature branch workflow**

Основная идея модели **feature branch workflow** (от англ. *feature* — «особенность», «деталь») в том, что работа над любой новой функциональностью или исправлением ведётся в отдельной ветке.

Допустим, у вас есть задача добавить большую оранжевую кнопку. Тогда вам нужно будет создать ветку feature/add-big-orange-button. Ветка может называться и по-другому — главное, чтобы это была отдельная ветка под конкретную задачу. Все коммиты, которые относятся к созданию кнопки, попадают в эту ветку. Когда кнопка полностью готова, ветка feature/add-big-orange-button вливается (merge) в ветку main.



Подход feature branch workflow позволяет разработчикам трудиться над новыми функциональностями (или **фичами**) независимо друг от друга и не затрагивать основную ветку кода в процессе. Так как каждый работает над своей веткой самостоятельно, не возникает проблемы «расходящейся» общей ветки, которую мы описали в прошлом уроке.

При таком подходе все коммиты в ветке main — это коммиты слияния (feature-ветки вливаются в main).

Если аккуратно следовать подходу feature branch workflow, то:

* Не будет проблемы «расхождения» веток, ведь новые изменения попадают в main через git merge, а не через git push. Команду merge «разошедшиеся» ветки не смущают, ведь для них она и придумана.
* В ветке main всегда рабочая версия проекта. Все «полуфабрикаты» и недоделанные функциональности находятся в feature-ветках, пока не будут готовы попасть в main.

### **О feature branch workflow коротко**

Основные правила:

* новая функциональность или исправление — новая ветка;
* когда код в feature-ветке готов, он вливается в main;
* в main всегда рабочая версия без «недоделок».

Преимущества:

* простая модель;
* позволяет работать с Git в команде без лишних технических сложностей.

Подход feature branch workflow — это только «заготовка», которой необязательно следовать строго. Каждая команда подстраивает процесс под себя: например, что-то добавляет в модель или убирает из неё.

Так что описать все существующие возможности подхода было бы сложно. Важно запомнить его ключевую особенность: одна ветка main и много feature-веток, которые вливают в основную.

Супер! Теперь вы знаете о подходах к работе с ветками немного больше. А именно:

* Разные компании и команды используют разные модели работы с Git. Эти модели описывают структуру веток в проекте, а также правила создания и слияния коммитов в них.
* Один из самых простых и популярных подходов — feature branch workflow. Он предполагает, что работа над функциональностью ведётся в отдельной feature-ветке. Когда всё готово, эта ветка вливается в основную.
* Подход feature branch workflow — лишь шаблон, по которому можно действовать. Многие компании и команды меняют его под свои нужды, но неизменно одно: хочешь сделать новую функциональность или исправить баг — создай новую ветку!