Работа с ветками в Git:

1. **Создание и просмотр веток:**
   - `git branch`
     Показывает список всех локальных веток. Текущая ветка выделена звездочкой (`*`).
   - `git branch <имя-ветки>`
     Создаёт новую ветку с указанным именем.
   - `git branch -v`
     Показывает список веток с краткой информацией о последнем коммите в каждой ветке.
   - `git branch -d <имя-ветки>`
     Удаляет локальную ветку, если она была слита. Для принудительного удаления используется флаг `-D`.
   - `git checkout <имя-ветки>`
     Переключает на указанную ветку. Эта команда обновляет рабочую директорию содержимым ветки.
   - `git checkout -b <имя-ветки>`
     Создаёт новую ветку и сразу переключается на неё.

Объединение веток:

2. `git merge <ветка>`
   Объединяет указанную ветку с текущей. Если нет конфликтов, изменения сливаются автоматически. Если конфликты есть, нужно их решить вручную и завершить слияние командой `git merge --continue`.
   - **Fast-forward** — автоматическое слияние, если нет отклонений, и история линейна.
   - **Non-fast-forward** — создаёт новый коммит слияния для объединения изменений, если в обеих ветках есть собственные коммиты.

Обновление ветки по мастеру:

3. `git rebase <ветка>`
   Перемещает текущую ветку на основе изменений другой ветки. Например, для обновления рабочей ветки по последним изменениям из `master` или `main`, делаем: `git rebase main`.
   После ребейза ветка становится как будто "сверху" основного дерева коммитов, сохраняя линейную историю, без коммитов слияния.

Выборочные изменения из другой ветки:

4. `git cherry-pick <хэш-коммита>`
   Копирует указанный коммит из другой ветки в текущую, не сливая полностью ветку. Это удобно, когда нужно забрать только одно изменение.

Комбинированное использование `git rebase` для веток:

5. `git rebase -i <хэш-коммита>`
   Интерактивный ребейз. Используется для редактирования истории или объединения коммитов. Можно выбирать, какие коммиты оставить, какие объединить (**squash**), а какие удалить (**fixup**). Полезно при работе с большими ветками.

Fast-forward и Non-fast-forward в Git:

6. **Fast-forward**
   Если текущая ветка полностью совпадает с другой, Git просто перемещает указатель ветки вперёд на основе изменений другой ветки, не создавая коммита слияния.
   Non-fast-forward
   Если в обеих ветках есть собственные коммиты, Git создаёт новый коммит слияния, который объединяет их изменения.