Для того чтобы научиться работать с несохранёнными данными и уметь перемещать их с одной ветки на другую без записи в рамках того, или иного коммита, используется функционал под названием Git Stash.   
Stash это специальное пространство, которое хранится у вас только на локальном компьютере и записывает все текущие изменения, отменяя состояния самого репозитория.

На ветке main изменяем файл dev.md  
Допустим мы не хотим коммитить эти изменения и хотим перейти на другую ветку.  
Для этого мы прописываем “git stash”. Мы получаем сообщение, что текущая рабочая директория и индекс будут перемещены в специальное пространство (work in progress). То есть на ветке main теперь записывается специальный stash commit который не попадает в общую историю, но при этом может быть восстановлен.  
Для проверки доступных текущих stash’ев команда “git stash list”.

Теперь можем перейти на другую ветку.   
Здесь также изменим этот же файл и при проверке git status мы видим, что он попал в список modified, но при этом, он не попадает в конфликт с предыдущей веткой. Что бы перейти обратно на ветку main, нам необходимо также   
отправить их в stash.

Если мы хотим в уже существующий stash добавить какой-то новый файл (у него должно быть изменено его содержимое), мы должны написать следующую команду:  
git stash push [название файла]  
Данная команда добавит файл к уже существующему stash’у на текущей ветке.   
Если не указывать push и название файла, он будет создаваться новым.

Команда чтобы применить изменения из stash’a  
“git stash pop [номер stash’а]”

Squash или склеивание коммитов.

Операция squash.  
Для показа этой операции мы создадим новую ветку и новый файл в ней, добавим коммит.  
Затем изменим этот файл, и добавим второй коммит. Снова изменим этот файл, и добавим третий коммит. Есть два способа объединить эти три коммита.  
Первый: это merge с указанием параметра squash. Для этого переключаемся на ветку main;  
прописываем команду “git merge [название ветки] --squash”. Если мы просмотрим git status, то увидим, что у нас создан новый файл, в котором записаны изменения третьего коммита(на картинке не уместилось)  
Таким образом, этот способ работает так, что он подтягивает изменения с указанной ветки, но не создаёт коммит объединения, для того что бы вы могли записать все изменения в рамках одного коммита

Операция squash.  
Второй способ.  
Вернемся на ветку sqгash-dev (ту ветку, которую вы создавали для операции squash). Теперь наша задача объединить все три коммита в рамках одного, и его в последствии в merge’ть в ветку main. Для того что бы это сделать, мы воспользуемся командой “git rebase –i HEAD~3”  
(-i что бы запустить в интерактивном режиме), (3 – количество коммитов которые вы хотите объединить). В открывшемся редакторе нужно заменить первое слово во второй и последующих строках, на букву f или fixup (для редактирования нужно нажать англ. A).

Cherry Pick.   
Это специальная команда, которая позволяет скопировать изменения коммита, и записать новый коммит на ваши текущие ветки.

Создадим новую ветку cherry. На ней используем команду “git cherry [хэш коммита]”. (для того, что бы посмотреть коммиты со всех веток, команда “git log --oneline --all”.

Перезапись коммита (amend). В случае, если вы только что сделали коммит, и поняли, что допустили маленькую ошибку или просто указали не то содержимое указали в файле, мы можем исправить эту ситуацию, добавив к своему коммиту специальный аргумент.

Для перезаписи коммита создаём файл fail.md, индексируем его и добавляем коммит. Предположим, что по задумке, мы должны были добавить текст в этот файл, и только потом сделать коммит. Для этого и нужен amend. Мы добавляем текст, снова индексируем и пишем команду “git commit --amend”. Обратите внимание, в логах у одного и того же коммита разные индексы, следовательно, в случае необходимости мы можем сделать remote по первоначальной версии коммита.

Решение конфликтов.  
Конфликт – это когда два разработчика изменили один и тот же файл, и попытались потом объединить его в master(main) или свои ветки. Такое часто бывает при работе над одним функционалом параллельно.

Создадим от ветки main две ветки dev1 и dev2. В этих ветках в файле “new-feature.go” напишем разный текст. Сделаем коммиты. Теперь переключимся на ветку main.

Начинаем слияние веток dev1 и dev2 в ветку main, используя команду “git merge dev1 dev2”. Терминал пишет следующее: “Попытка объединения dev1 и dev2 с помощью simple merge (обычный merge);  
Simple merge не сработал;  
Попытка с помощью auto-merging (авто merge);  
auto-merging получил конфликты, конфликты должны решаться ручным воздействием”. При этом у нас не до конца отменяется операция merge. Это можно увидеть при просмотре “git status”.

Для решения этого конфликта, мы можем открыть через текстовый редактор (например “vi [название файла]”) файл “new-feature.go”. В редакторе, мы меняем текст так, как показано на второй картинке. Добавляем индекс и делаем коммит.