УРОК 12. MERGE И REBASE

ЧТО ТАКОЕ GIT MERGE

Git merge - это операция в системе контроля версий Git, которая объединяет изменения из одной ветки с другой. Когда вы выполняете слияние веток, Git автоматически объединяет изменения, сделанные в одной ветке, с изменениями из другой ветки.

Git merge используется для объединения "целевой" ветки с веткой "исходной". Git анализирует изменения, сделанные в каждой из этих веток, и пытается автоматически объединить их.

Самым простым вариантом объединить main и feature branch:

```
Unset
git checkout feature
git merge main
```

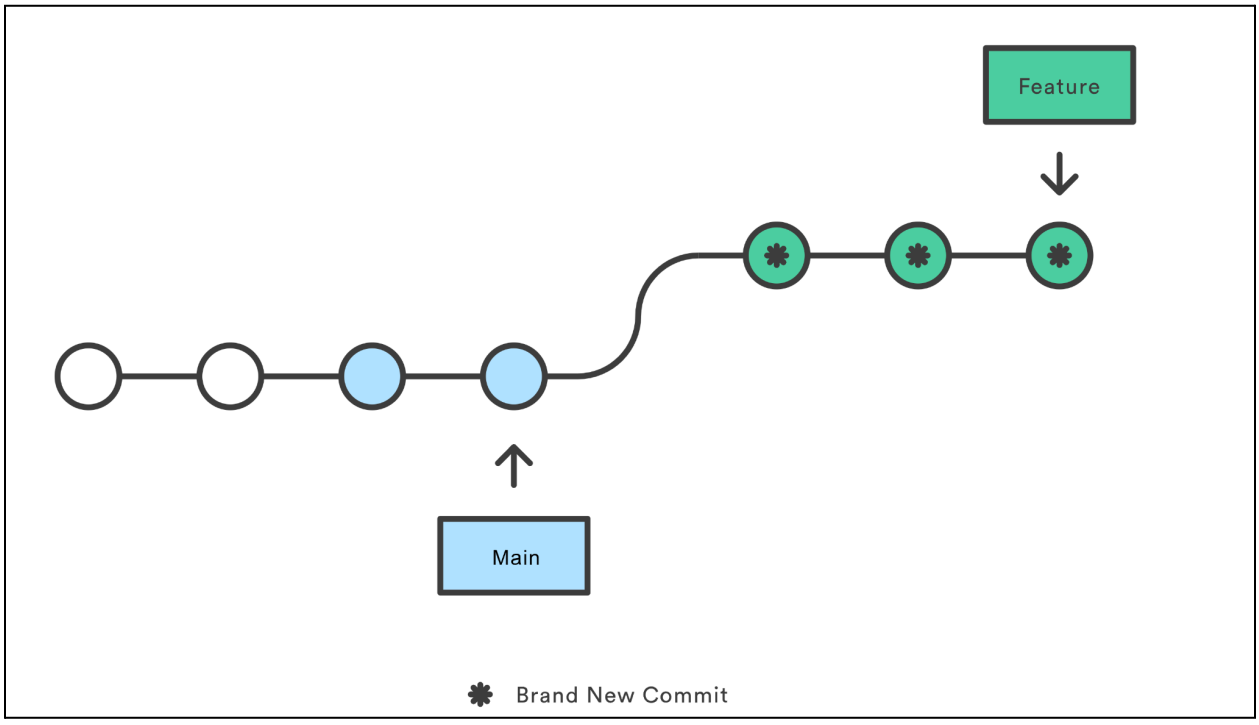
Или в одну строку:

```
Unset
git merge feature main
```

ЧТО ТАКОЕ GIT REBASE

Git rebase - это операция в системе контроля версий Git, которая позволяет перебазировать (переместить) коммиты одной ветки на другую ветку.

Git rebase, Git берет коммиты из вашей текущей ветки и применяет их поверх коммитов другой ветки (чаще всего это ветка, от которой вы решили отойти с помощью ребейза). Это позволяет поддерживать более линейную историю коммитов без лишних слияний.



В качестве альтернативы merge можно перебазировать (rebase) feature branch в main branch:

```
Unset
git checkout feature
git rebase main
```

Чем отличаются git rebase и git merge:

История коммитов:

- Merge: Создает дополнительный коммит слияния, который объединяет изменения из двух веток. Этот коммит слияния сохраняет историю обеих веток.
- Rebase: Переписывает историю коммитов путем перемещения (или "перебазирования") коммитов текущей ветки поверх целевой ветки. В результате создаются новые коммиты с теми же изменениями, но примененными к целевой ветке. История коммитов становится линейной.

Чистота истории:

- Merge: В истории коммитов сохраняются все слияния, что может привести к "мусорным" коммитам слияния.
- Rebase: История коммитов становится более чистой и линейной, так как все коммиты текущей ветки применяются поверх целевой ветки без создания дополнительных коммитов слияния.

Конфликты:

- Merge и Rebase: Конфликты могут возникнуть при слиянии двух веток и при применении коммитов текущей ветки к целевой ветке, и их придется разрешить вручную.

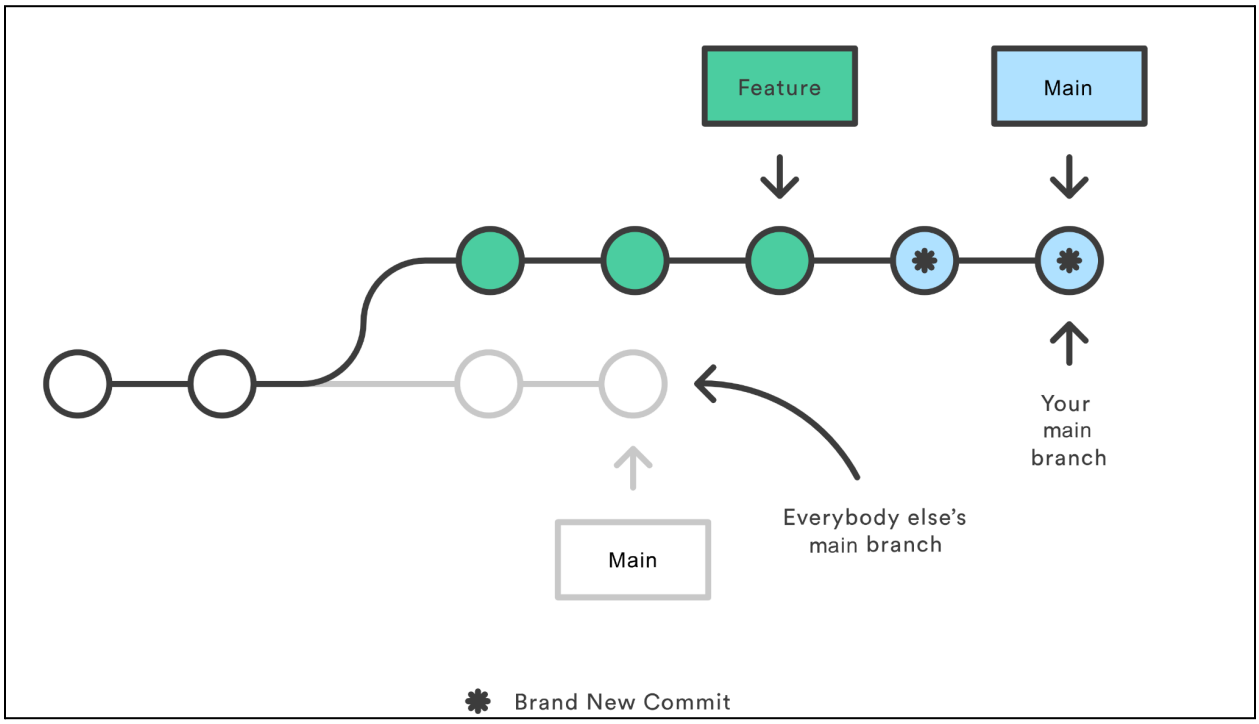
Безопасность - “Проблемность”:

- Merge: Более безопасен, поскольку не изменяет историю коммитов.
- Rebase: Может быть опасен при работе с общедоступными ветками, так как изменяет историю коммитов, что может привести к потере данных и конфликтам при совместной работе.

ЗОЛОТОЕ ПРАВИЛО REBASE

Итак, мы разобрались, что такое перебазирование. Теперь главное - понять, когда его делать не нужно.

Золотое правило перебазирования в Git заключается в том, что его никогда не следует применять к (публичным) общедоступным веткам.



Rebase перемещает все коммиты из основной ветки в feature ветку.

Проблема заключается в том, что это происходит только в вашем репозитории. Все остальные разработчики все еще работают с исходной основной веткой. Поскольку перебазирование приводит к созданию совершенно новых коммитов, Git будет считать, что история вашей основной ветки расходится с историей всех остальных.

Единственный способ синхронизировать две основные ветки - это снова слить их вместе, что приведет к дополнительному коммиту слияния и двум наборам коммитов, содержащих одни и те же изменения (оригинальные и те, из вашей перебазированной ветки).

Прежде чем запустить git rebase, всегда задайте себе вопрос: "Кто-нибудь еще работает с этой веткой ?" Если ответ утвердительный - не делайте этого!

Если вы попытаетесь отправить перебазированную основную ветку обратно в удаленный репозиторий, Git не позволит вам это сделать. Однако вы можете принудительно выполнить отправку, с помощью --force:

```
Unset

git push --force
```

Это перезапишет основную ветку. Будьте осторожны!

GIT COMMIT –-AMEND

Git commit --amend - это команда, которая используется для изменения последнего коммита в репозитории и позволяет внести изменения в последний коммит без создания нового коммита.

Команда очень удобна для исправления небольших ошибок или добавления упущенных изменений в последний коммит.

После выполнения этой команды откроется редактор сообщения коммита, где вы можете внести изменения в сообщение коммита, если это необходимо.

Также вы можете добавить новые файлы в индекс перед выполнением git commit --amend, и они будут включены в исправленный коммит.

Важно помнить, что изменения, внесенные с помощью git commit --amend, изменят историю коммитов в вашем локальном репозитории. Поэтому, если вы уже поделились этим коммитом с другими, будьте осторожны, чтобы не создать проблем с совместной работой.

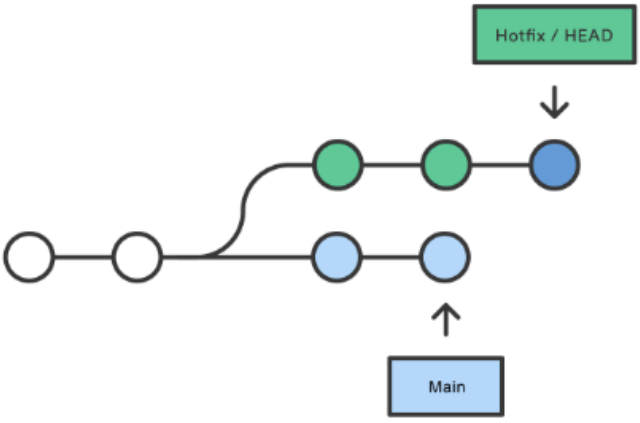
GIT RESET

Git reset используется для изменения состояния вашего репозитория. Эта команда изменяет указатель HEAD и/или перемещает текущую ветку на определенный коммит.

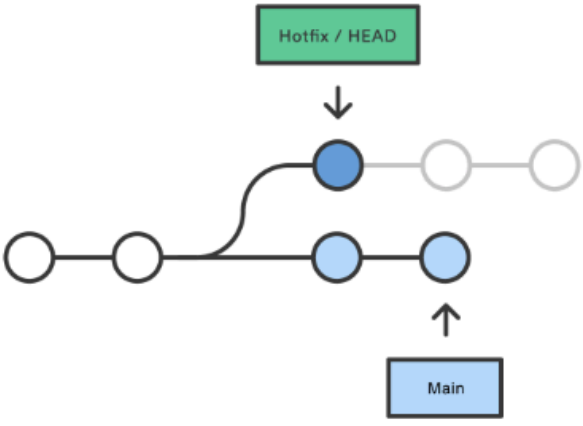
Основные опции git reset включают:

- --soft: Сбрасывает HEAD на указанный коммит, оставляя изменения в рабочем каталоге и индексе без изменений. Это позволяет вам "отменить" коммиты, не теряя внесенные изменения.
- --mixed (по умолчанию): Сбрасывает HEAD на указанный коммит и сбрасывает индекс, сохраняя изменения в рабочем каталоге. Это используется, чтобы отменить индексацию изменений после предыдущих коммитов.
- --hard: Сбрасывает HEAD на указанный коммит, сбрасывает индекс и откатывает все изменения в рабочем каталоге. Будьте осторожны, так как эта команда безвозвратно удаляет все непроиндексированные изменения.

Before Resetting



After Resetting



○ - Orphaned Commits

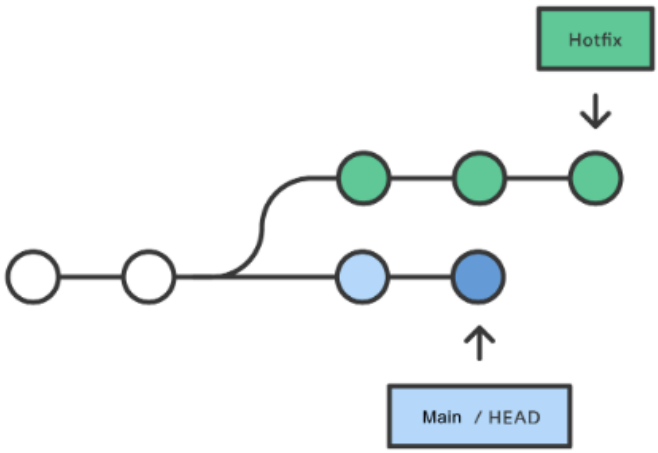
GIT CHECKOUT

Git checkout используется для переключения между ветками и восстановления файлов из репозитория.

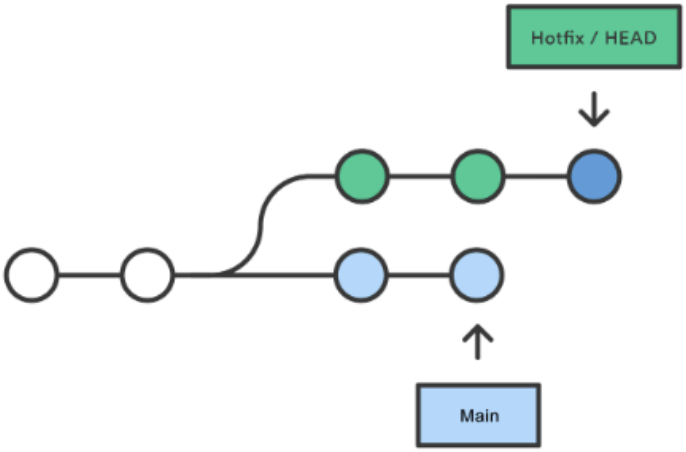
Основные использования git checkout включают:

- Переключение между существующими ветками с использованием их имен.
- Создание новых веток на основе существующих веток или коммитов.
- Восстановление файлов из определенного коммита, а не из последней версии ветки.

Before Checking Out



After Checking Out

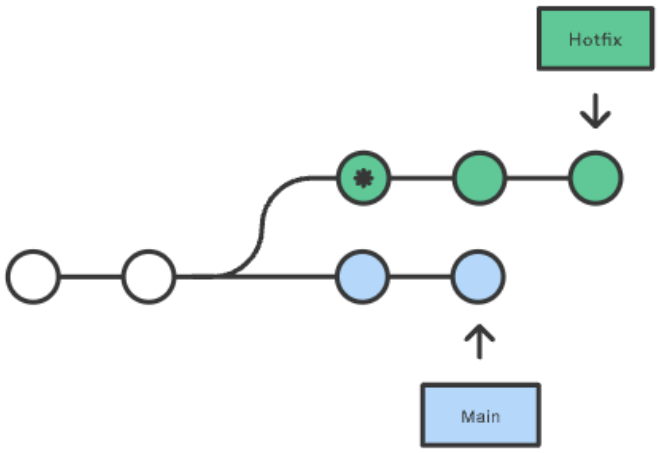


GIT REVERT

Git revert используется для отмены изменений, внесенных определенным коммитом, путем создания нового коммита, который отменяет изменения этого коммита.

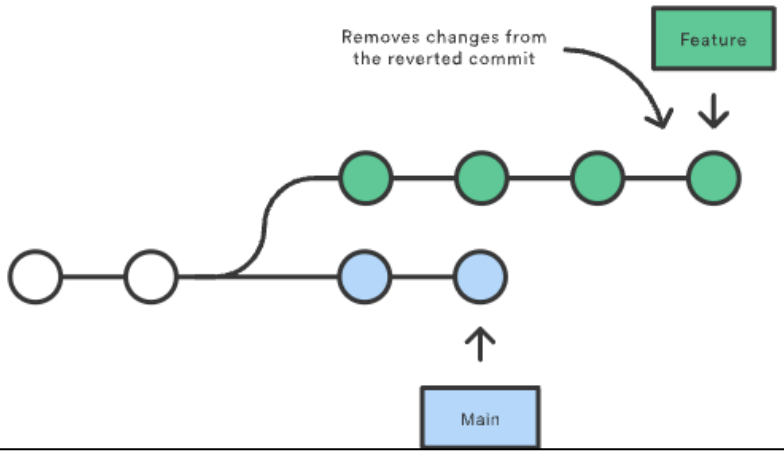
Git revert безопасен для использования в общедоступных ветках, так как он не изменяет историю коммитов, а создает новые коммиты, отменяющие изменения.

Команда git revert полезна, когда вы хотите откатить изменения, но не хотите изменять историю коммитов, особенно если другие разработчики уже получили доступ к изменениям.



Commit to be reverted

After Reverting



Отличия revert от reset:

Основное различие между ними заключается в том, что revert создает новый коммит для отмены изменений, в то время как reset изменяет состояние репозитория, не создавая новые коммиты. Кроме того, revert безопаснее для использования в общедоступных ветках, так как сохраняет историю коммитов, в то время как reset может быть опасным и привести к потере данных.

git revert:

- Создает новый коммит, который отменяет изменения, внесенные определенным коммитом.
- Безопасен для использования на общедоступных ветках, так как не изменяет историю коммитов.
- Полезен, когда вы хотите отменить изменения, сохраняя при этом историю коммитов и уведомляя других разработчиков о внесенных изменениях.

git reset:

- Изменяет состояние вашего репозитория путем перемещения указателя HEAD и/или текущей ветки на определенный коммит.
- Используется для отмены изменений, но без создания новых коммитов.
- Может быть опасным для использования на общедоступных ветках, так как изменяет историю коммитов и может привести к потере данных для других разработчиков.