ВЕТВЛЕНИЕ И ПРОДВИНУТАЯ РАБОТА С GIT

Ветка Branch

отдельная линия разработки в репозитории, которая позволяет изолировать изменения и нововведения до момента их слияния с основной линией (например, main или master)

Основная ветка Main branch

главная линия разработки, обычно используемая для стабильной версии проекта. Часто называется main или master

---

Новая ветка Feature branch

ветка, которая создается для разработки новой функциональности или исправления ошибок. Часто называют feature-branch, используется для изоляции изменений до их слияния с основной веткой

---

stash

команда в Git, которая позволяет временно сохранить незафиксированные изменения в локальном репозитории. Это полезно, когда нужно переключиться на другую ветку, не теряя текущих изменений

---

git stash apply

команда, которая позволяет восстановить ранее сохраненные изменения из хранилища stash, чтобы продолжить работу над ними

Detached HEAD

состояние в Git, когда HEAD указывает на определенный коммит, а не на ветку. Это может произойти при переходе к конкретному коммиту с использованием хеша

HEAD

указатель на текущий коммит или ветку, активно работающие в репозитории

Слияние Merge

процесс объединения изменений из одной ветки в другую. Обычно применяется для интеграции изменений из feature-ветки в основную ветку для подготовки к выпуску новой версии

Перебазирование Rebase

альтернатива слиянию, при которой изменения из одной ветки переносятся на другую с сохранением линейной истории. Это помогает сохранить чистоту истории изменений

Конфликт слияния Merge Conflict

ситуация, когда автоматическое слияние веток невозможно из-за конфликтующих изменений в одних и тех же частях кода

Разрешение конфликта Conflict Resolution

процесс ручного вмешательства разработчика для выбора верных изменений в случае конфликта. После разрешения необходимо зафиксировать изменения

Удаление локальной ветки

процесс удаления ветки из локального репозитория после завершения работы над ней. Осуществляется командой git branch -d или git branch -D для форсированного удаления

Удаление удаленной ветки

процесс удаления ветки из удаленного репозитория. Выполняется командой git push origin –delete branch-name, устраняя ненужные или устаревшие ветки

Защищенные ветки Protected Branches

настройки в системах контроля версий (например, GitHub), которые ограничивают возможность изменения определенных веток, таких как основная, для предотвращения случайных изменений

Чистка веток Branch Cleanup

регулярное удаление нерелевантных или устаревших веток для поддержания порядка в репозитории и упрощения управления проектом

---

git log

основная команда для просмотра истории коммитов. Может быть дополнена различными опциями, такими как –oneline для краткого представления или –graph для визуализации ветвления

---

git reflog

позволяет отслеживать перемещения указателя HEAD, предоставляя доступ к истории всех изменений, даже если они больше не видны в обычной истории коммитов

Удаление коммитов

использование команд вроде git reset для полного удаления коммитов из локальной истории. Важно помнить, что удаленные коммиты могут быть восстановлены через reflog, если не было жесткой чистки

Чистка мусора git gc

команда для очистки ненужных файлов и ликвидации мусора, которая может совмещаться с такими командами, как git prune для удаления висячих объектов

Полное удаление репозитория

удаление папки .git из проекта, что приводит к полной потере всей истории версии

Объекты Git

основные структурные единицы, такие как blob (содержимое файлов), tree (каталоги) и commit (история изменений), которые представляют собой снимки состояния репозитория на каждый момент времени

Индекс Staging Area

промежуточная область, где изменения подготавливаются к коммиту, позволяет более точно контролировать, какие изменения включаются в каждый коммит

Указатели References

ссылки на коммиты, включая ветки и теги, которые позволяют легко управлять различными состояниями проекта

git reset

команда для изменения состояния текущей ветки, позволяющая «откатить» изменения и передвинуть HEAD на другой коммит. Поддерживает варианты –soft, –mixed, и –hard, которые определяют, какие изменения сохраняются в рабочем дереве и индексе

---

git revert

откатывает изменения, создавая новый коммит, который противоположен указанному. Этот метод считается более безопасным в общих репозиториях, поскольку сохраняет историю изменений

---

git stash

временное сохранение изменений, которые не готовы для коммита, чтобы вернуться к чистому состоянию рабочей директории, с возможностью восстановления изменений позже