Содержание

git

доки

интерактивная обучалка

о комментариях к коммиту

Представьте, что вы пишете код и внезапно понимаете, что предыдущая версия работала лучше. Или вы работаете с командой, и нужно объединить изменения нескольких человек. Git решает эти задачи, сохраняя историю всех изменений и позволяя безопасно экспериментировать с кодом.

Git хранит снимки вашего проекта в специальном хранилище, которое называется **репозиторием**. Каждое сохранение изменений называется **коммитом**, и вы всегда можете вернуться к любому из них.

Настройка

Создание репозитория

```
# Создаёт новый локальный репозиторий с заданным именем
git init <название проекта>

# Скачивает репозиторий вместе со всей его историей изменений
git clone <url-адрес>
```

Внесение изменений

```
# Перечисляет все новые или изменённые файлы, которые нуждаются в фиксации
git status

# Показывает различия по внесённым изменениям в ещё не проиндексированных файлах
git diff

# Индексирует указанный файл для последующего коммита
git add <файл>

# Показывает различия между проиндексированной и последней зафиксированной версиями файлов
git diff --staged
```

```
# Отменяет индексацию указанного файла, при этом сохраняет его содержимое
git reset <файл>

# Фиксирует проиндексированные изменения и сохраняет их в историю версий
git commit -m "<сообщение с описанием>"
```

Операции с файлами

```
# Удаляет конкретный файл из рабочей директории и индексирует его удаление
git rm <файл>

# Убирает конкретный файл из контроля версий, но физически оставляет его на своём месте
git rm --cached <файл>

# Перемещает / переименовывает указанный файл, сразу индексируя его для последующего коммита
$ git mv <оригинальный файл> <новое имя>
```

Просмотр истории

```
# История коммитов для текущей ветки
git log

# История изменений конкретного файла, включая его переименование
git log --follow <файл>

# Показывает разницу между содержанием коммитов двух веток
git diff <первая ветка>...<вторая ветка>

# Выводит информацию и показывает изменения в выбранном коммите
git show <коммит>
```

Фрагменты

```
# Временно сохраняет все незафиксированные изменения отслеживаемых файлов
git stash
git stash push -m "имя какое тебе удобно для обозначения твоих изменения"

# Восстанавливает состояние ранее сохранённых версий файлов
git stash pop
git stash apply stash@{<тут номер стэша>}

# Выводит список всех временных сохранений
git stash list
```

```
# Сбрасывает последние временно сохранённые  изменения
git stash drop
```

Branching

```
# Список именованных веток коммитов с указанием выбранной ветки
git branch

# Создание ветки
git branch имя_ветки
# удаляет выбранную ветку
git branch -d <имя ветки>

# Переключение активной ветки
git checkout имя_ветки

# Слияние веток
git checkout master # входим в ветку, которую хотим слить
git merge <имя_ветки> # сливаем с другой веткой
```

Remotes

```
# Добавляет новый удалённый репозиторий с указанным именем
git remote add <имя> <URL>

# Удаляет удалённый репозиторий с указанным именем
git remote remove <имя>

# Переименовывает удалённый репозиторий с указанным старым именем на новое имя
git remote rename <старое_имя> <новое_имя>

# Обновляет  URL  удалённого репозитория
git remote set-url <имя> <новый_URL>

# Показывает список всех удалённых репозиториев
git remote -v

# Показывает подробную информацию о конкретном удалённом репозитории
git remote show <имя>

# Скачивает всю историю из удалённого репозитория
git fetch <удалённый репозиторий>

# Вносит изменения из ветки удалённого репозитория в текущую ветку локального
репозитория
git merge <удалённый репозиторий>/<ветка>
```

```
# Загружает все изменения локальной ветки в удалённый репозиторий
git push <удалённый репозиторий> <ветка>

# Загружает историю из удалённого репозитория и объединяет её с локальной. pull = fetch + merge
git pull
```

Merge

Fix Me!

Tegging

Fix Me!

Rewrite history

Удаление ошибок и корректировка созданной истории

```
# Отменяет все коммиты после заданного, оставляя все изменения в рабочей директории
git reset <коммит>

# Сбрасывает всю историю вместе с состоянием рабочей директории до указанного коммита.
git reset --hard <коммит>

# Отмена последнего коммита
# На самом деле это несовсем отмена коммита, а создание нового, идентичного пред-предыдущему
git revert HEAD
# Отмена конкретного коммита
git revert <хэш_коммита>
```