Установка Git

GitHub предоставляет оконное приложение с графическим интерфейсом для выполнения основных операций с репозиторием, и консольную версию Git с автоматическими обновлениями для расширенных сценариев работы.

GitHub Desktop

desktop.github.com

Дистрибутивы Git для систем Linux и POSIX доступны на официальном сайте Git SCM.

Git для всех платформ

git-scm.com

Первоначальная настройка

Настройка информации о пользователе для всех локальных репозиториев

```
$ git config --global user.name "[имя]"
```

Устанавливает имя, которое будет отображаться в поле автора у выполняемых вами коммитов

```
$ git config --global user.email "[адрес электронной почты]"
```

Устанавливает адрес электронной почты, который будет отображаться в информации о выполняемых вами коммитах

Создание репозитория

Создание нового репозитория или получение его по существующему URL-адресу

```
$ git init [название проекта]
```

Создаёт новый локальный репозиторий с заданным именем

```
$ git clone [url-адрес]
```

Скачивает репозиторий вместе со всей его историей изменений

Операции с файлами

Перемещение и удаление версий файлов репозитория

```
$ git rm [файл]
```

Удаляет конкретный файл из рабочей директории и индексирует его удаление

```
$ git rm --cached [файл]
```

Убирает конкретный файл из контроля версий, но физически оставляет его на своём месте

```
$ git mv [оригинальный файл] [новое имя]
```

Перемещает и переименовывает указанный файл, сразу индексируя его для последующего коммита

Игнорирование некоторых файлов

Исключение временных и вторичных файлов и директорий

```
*.log
build/
temp-*
```

Git будет игнорировать файлы и директории, перечисленные в файле `.gitignore` с помощью wildcard синтаксиса

```
$ git ls-files --others --ignored --exclude-standard
```

Список всех игнорируемых файлов в текущем проекте

Сохранение фрагментов

Сохранение и восстановление незавершённых изменений

```
$ git stash
```

Временно сохраняет все незафиксированные изменения отслеживаемых файлов

```
$ git stash pop
```

Восстанавливает состояние ранее сохранённых версий файлов

```
$ git stash list
```

Выводит список всех временных сохранений

```
$ git stash drop
```

Сбрасывает последние временно сохранённые изменения

Внесение изменений

Просмотр изменений и создание коммитов (фиксация изменений)

```
$ git status
```

Перечисляет все новые или изменённые файлы, которые нуждаются в фиксации

```
$ git diff
```

Показывает различия по внесённым изменениям в ещё не проиндексированных файлах

```
$ git add [файл]
```

Индексирует указанный файл для последующего коммита

```
$ git diff --staged
```

Показывает различия между проиндексированной и последней зафиксированной версиями файлов

```
$ git reset [файл]
```

Отменяет индексацию указанного файла, при этом сохраняет его содержимое

```
$ git commit -m "[сообщение с описанием]"
```

Фиксирует проиндексированные изменения и сохраняет их в историю версий

Коллективная работа

Именованные серии коммитов и соединение результатов работы

```
$ git branch
```

Список именованных веток коммитов с указанием выбранной ветки

```
$ git branch [имя ветки]
```

Создаёт новую ветку

```
$ git switch -c [имя ветки]
```

Переключается на выбранную ветку и обновляет рабочую директорию до её состояния

```
$ git merge [имя ветки]
```

Вносит изменения указанной ветки в текущую ветку

```
$ git branch -d [имя ветки]
```

Удаляет выбранную ветку

Просмотр истории

Просмотр и изучение истории изменений файлов проекта

```
$ git log
```

История коммитов для текущей ветки

```
$ git log --follow [файл]
```

История изменений конкретного файла, включая его переименование

```
$ git diff [первая ветка]...[вторая ветка]
```

Показывает разницу между содержанием коммитов двух веток

```
$ git show [коммит]
```

Выводит информацию и показывает изменения в выбранном коммите

Откат коммитов

Удаление ошибок и корректировка созданной истории

```
$ git reset [коммит]
```

Отменяет все коммиты после заданного, оставляя все изменения в рабочей директории

```
$ git reset --hard [коммит]
```

Сбрасывает всю историю вместе с состоянием рабочей директории до указанного коммита.

Синхронизация с удалённым репозиторием

Регистрация удалённого репозитория и обмен изменениями

```
$ git fetch [удалённый репозиторий]
```

Скачивает всю историю из удалённого репозитория

```
$ git merge [удалённый репозиторий]/[ветка]
```

Вносит изменения из ветки удалённого репозитория в текущую ветку локального репозитория

```
$ git push [удалённый репозиторий] [ветка]
```

Загружает все изменения локальной ветки в удалённый репозиторий

```
$ git pull
```

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