💡 Основы Git

Введение

GIT - распределённая система контроля версий (VCS - Version Control System)

Существует 3 типа систем контроля версий:

- Локальная - самая простая система. Создание локальной резервной копии, копируя файлы.
- Централизованная - существует центральный сервер, где хранится последняя версия каждого файла.
- Распределённая - каждый разработчик имеет полную копию репозитория, включая всю историю изменений

Git Configuration

Для просмотра полной Git конфигурации

```
git config --list --show-origin
```

Для изменения конфигурации пользователя на глобальном уровне

```
git config --global user.name Viacheslav
git config --global user.email v.a.asadchiy@gmail.com

git config --global --unset user.name
git config --global --unset user.email
```

Для изменения конфигурации пользователя на уровне проекта

```
git config --local user.name Viacheslav
git config --local user.email v.a.asadchiy@gmail.com

git config --local --unset user.name
git config --local --unset user.email
```

Полное удаление секции из `.git`

```
git config --local --remove-section user
```

Основные комманды

Для получения справки по коммандам

```
git help <command>
```

Доступные комманды

```
git help -a
```

Инициализация репозитория

```
git init
```

Текущее состояние репозитория

```
git status
```

Добавить файл для отслеживания (переместить в промежуточную область - индекс staging area)

```
git add
```

Зафиксировать все проиндексированные файлы

```
git commit -m "Краткое описание изменений"
```

При совершении коммита генерируется уникальный хэш код SHA1

Для просмотра всех коммитов и краткой информации о них

```
git log
```

Отслеживание коммитов и изменений

Для отслеживания разницы в файле используется команда `diff` которая приимает два входных файла и показывает разницы
Для проверки разницы между рабочей областью и индексом

```
git diff
```

Для проверки разницы между индексом и локальным репозиторием

```
git diff --staged
```

Для проверки разницы между рабочей областью и локальным репозиторием

```
git diff head
```

Git не хранит просто файлы или их отличия (diff), как это делают другие VCS.
Git сохраняет снимки (snapshots) содержимого и представляет всё в виде объектов, связанных SHA-1 хешами.
Git использует SHA-1 хеш-функцию

```
e83c5163316f89bfbde7d9ab23ca2e25604af290
```

В Git всё построено на 4 типах объектов:

Тип объекта	Назначение

Blob	Содержимое файла (без имени, только данные)
Tree	Каталог (ссылки на blob'ы и другие tree'и)
Commit	Снимок проекта с метаданными и ссылкой на tree
Tag	Указатель на commit (или объект), с метаданными

*Все это хранится в **objects** папке*

Посмотреть содержимое объекта по хешу (pretty-print).

```
git cat-file -p <hash>
```

Восстановление и переименование

В случае изменения имени файла git считает что мы удалили старый файл и создали новый.

```
mv <old_name.txt> <new_name.txt>
git add <new_name.txt>
git rm <old_name.txt>
```

Поэтому существует команда git mv

```
git mv <old_name.txt> <new_name.txt>
```

Для восстановления файла из последнего коммита (HEAD)

```
git restore <file.txt>
```

Чтобы восстановить стейдженные изменения (из индекса)

```
git restore --staged <file.txt>
```

Это удалит файл из индекса, но оставит изменения в рабочей директории (удобно при ошибочном **git add**).