ИСПОЛЬЗОВАНИЕ GIT В КОМАНДНОЙ СТРОКЕ WINDOWS

УСТАНОВКА И НАСТРОЙКА

Для начала необходимо установить сам Git для Windows, сделать это можно с официального сайта. После успешной установки необходимо настроить Git, а именно ввести имя, почту, а также установить ветку по умолчанию для новых репозиториев. Все это можно сделать с помощью команды *config* (фрагмент кода 1).

```
# Имя пользователя
git config --global user.name «Имя»

# Почта пользователя
git config --global user.email «Почта»

# Ветка по умолчанию для новых репозиториев
git config --global init.defaultBranch (имя ветки)
```

Фрагмент кода 1. Первоначальная настройка

СОЗДАНИЕ РЕПОЗИТОРИЯ

Открыть новый репозиторий на компьютере можно двумя способами: клонирование существующего проекта с удаленного репозитория, открытие нового проетка локально. В первом случае можно воспользоваться командой *clone* в папке с проектом (фрагмент кода 2). Во втором случае нужно перейти в папку проекта и использовать команду *init* (фрагмент кода 3). Также можно узнать статус проекта с помощью команды status.

```
# Инициализация проекта
git clone (ссылка на удаленный репозиторий)
```

Фрагмент кода 2. Клонирование проекта из удаленного репозитория

```
# Инициализация проекта
git init

# Проверка статуса
git status
```

Фрагмент кода 3. Создание проекта локально

РАБОТА С ИЗМЕНЕНИЯМИ

Работа с Git происходит в следующем формате (фрагмент кода 4):

1. Проверка статуса файлов, чтобы узнать, какие из них были изменены (команда *status*)
2. Подготовка файлов к коммиту (сохранению изменений) с помощью команды *add*
3. Коммит файла (сохранение всех текущих изменений) с помощью команды *commit*

```
# Проверка статуса
git status

# Подготовка файла к коммиту
git add (имя файла)

# Подготовка всех файлов к коммиту
git add .

# Коммит файлов
git commit -m «Описание изменений»
```

Фрагмент кода 4. Сохранение всех изменений

Все изменения можно просмотреть с помощью команды *log*.

РАБОТА С ВЕТКАМИ

Над ветками можно производить следующие действия (фрагмент кода 5):

1. Создание ветки с помощью функции *branch*
2. Смена ветки через функцию *checkout*
3. Вывод списка из всех веток
4. Удаление ветки
5. Слияние веток с помощью функции *merge*

```
# Создание ветки
git branch (имя ветки)

# Переключение на другую ветку
git checkout (имя ветки)

# Вывод списка всех веток
git branch

# Удаление ветки
git branch -d branch-name

# Слияние веток (например, слияние brn в ветку main)
git checkout main
git merge brn
```

Фрагмент кода 5. Работа с ветками

РАБОТА С УДАЛЕННЫМ РЕПОЗИТОРИЕМ

С удаленным репозиторием можно производить следующие действия (фрагмент кода 6):

1. Вывод списка привязанных удаленных репозиториев
2. Добавление удаленного репозитория с помощью функции *remote*
3. Скачивание изменений с удаленного репозитория с помощью функции *fetch*
4. Отправка коммитов на удаленный репозиторий с помощью функции *push*

```
# Список всех привязанных удаленных репозиториев
git remote -v

# Добавление удаленного репозитория
git remote add (имя ветки) (ссылка на репозиторий)

# Скачивание изменений с удаленного репозитория
git fetch (имя ветки)

# Отправка коммитов на удаленный репозиторий
git push (имя ветки)
```

Фрагмент кода 6. Работа с удаленным репозиторием