

# Git & GitHub: Основы и первый рабочий процесс

Изучаем систему контроля версий: что такое Git, зачем он нужен разработчикам, и как начать работать с GitHub. Теория плюс практика для уверенного старта.

# Что такое Git



## Система контроля версий

Git отслеживает все изменения в ваших файлах, создавая полную историю проекта



## История изменений

Каждое изменение сохраняется как снимок, позволяя вернуться к любой версии



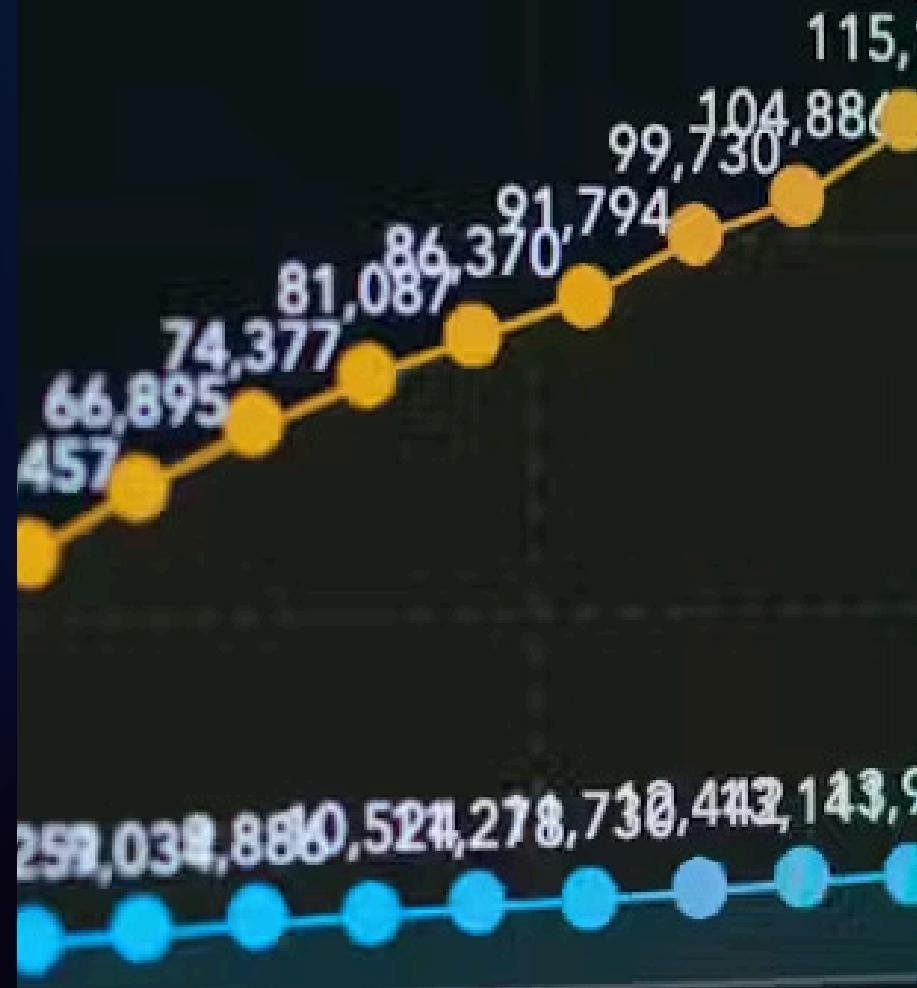
## Командная работа

Несколько разработчиков могут работать над одним проектом одновременно



## Откат изменений

Ошибки не страшны – всегда можно вернуться к рабочей версии



Apr



Confirmados

superior direito p

Made with GAMMA

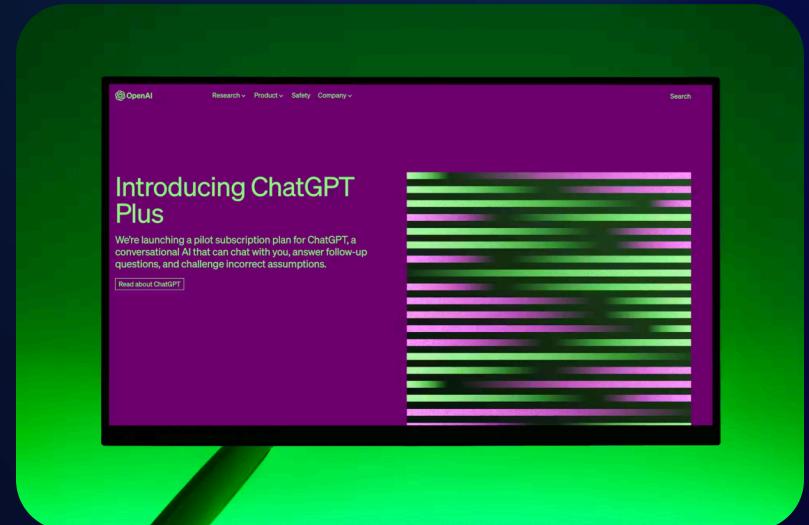
# Что такое GitHub

GitHub – это облачная платформа для хранения Git-репозиториев и совместной работы над кодом.

01

## Облачное хранилище

Ваши репозитории доступны из любой точки мира



02

## Pull Requests

Механизм для проверки и обсуждения изменений перед слиянием

03

## Issues

Система отслеживания задач и багов внутри проекта

04

## Аутентификация

Безопасный доступ через GitHub аккаунт

# Основные термины Git



## Repository (Repo)

Хранилище проекта со всей историей изменений



## Commit

Снимок изменений с описанием того, что было сделано



## Branch

Независимая линия разработки для работы над фичами



## Merge

Объединение изменений из одной ветки в другую



## Clone

Создание локальной копии удалённого репозитория



## Push / Pull

Отправка изменений на сервер и получение обновлений

# Начальная настройка Git

## Команды для первого запуска

```
git config --global user.name "Your Name"  
git config --global user.email  
"you@example.com"  
git config --list
```



### Зачем это нужно?

Каждый коммит должен содержать информацию об авторе изменений



### Где хранится?

Настройки сохраняются в файле .gitconfig в домашней директории



# Создание локального репозитория



Создать папку проекта

```
mkdir my_project  
cd my_project
```

Создаём директорию для нашего нового проекта



Инициализировать Git

```
git init
```

Превращаем обычную папку в Git-репозиторий



Проверить статус

```
git status
```

Просматриваем текущее состояние  
репозитория



# Первый коммит



## Создать файл

Создайте или измените файлы в рабочей директории

## Staging Area

```
git add .
```

Добавьте изменения в промежуточную область

## Commit

```
git commit -m "Initial commit"
```

Зафиксируйте изменения в локальном репозитории



**Важно:** Staging Area позволяет выбрать, какие изменения включить в коммит. Это дает контроль над историей проекта.

# Привязка к GitHub и первый push

01

## Создать репозиторий на GitHub

Зайдите на GitHub и создайте новый пустой репозиторий

02

## Привязать локальный репозиторий

```
git remote add origin https://github.com/user/repo.git  
git remote -v
```

03

## Отправить изменения

```
git push -u origin main
```



### origin

Стандартное имя для удалённого репозитория на GitHub

### main

Название главной ветки проекта (ранее master)



# Работа с ветками

## Команды для работы с ветками

```
git branch feature1
```

Создать новую ветку

```
git checkout feature1
```

Перейти на ветку

```
git switch -c feature1
```

Создать и перейти одной  
командой

```
git merge feature1
```

Объединить ветку с  
текущей

## Зачем нужны ветки?

- Разработка фич

Каждая новая функция  
разрабатывается в  
отдельной ветке

- Эксперименты

Пробуйте новые идеи  
без риска сломать  
основной код

- Минимизация  
конфликтов

Команда работает  
параллельно без  
блокировок

# Основной рабочий процесс

## Идеальный daily workflow разработчика

### Шаг 1: Получить изменения

`git pull`

Скачайте последние обновления с удалённого репозитория

### Шаг 2: Работа над задачей

Пишите код, создавайте и редактируйте файлы

### Шаг 3: Подготовка к коммиту

`git add .`

Добавьте все изменения в staging area

### Шаг 4: Создание коммита

`git commit -m "Описание изменений"`

Зафиксируйте изменения с понятным сообщением

### Шаг 5: Отправка на GitHub

`git push`

Загрузите коммит в удалённый репозиторий

### Шаг 6: Pull Request (опционально)

Создайте PR для ревью кода перед слиянием в main