

Лекция 1

# Знакомство с контролем версий





Видео лекции



Интерактивный учебник по git

Репозиторий, в котором есть примеры из лекции



1. Знакомство с контролем версий.pdf

## Цель лекции:

- \* Знакомство с контролем версий



## Таймкоды

1:00

Введение

3:03

Что такое контроль версий и зачем он нужен

14:35

Git - программа для контроля версий

26:59

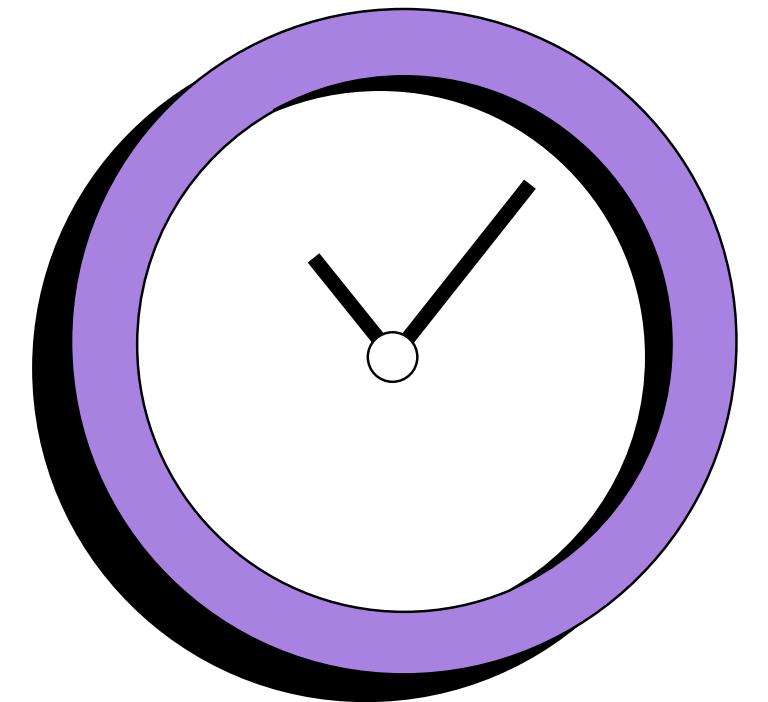
Команды Git

49:38

Синтаксис языка Markdown

1:11:35

Резюме синтаксиса Markdown и команд git



# Что такое контроль версий и зачем он нужен

**Контроль версий (контроль исходного кода)** — практика, которая позволяет отслеживать изменения исходного кода и управлять ими.

## Рассмотрим практическую задачу.

Мы создали сайт, он работает, пользователи не испытывают проблем. Но прошло время и мы решили изменить функциональность. Прежде чем вносить изменения, сохраняем рабочую версию сайта на компьютере или сервере. Это будет архив «Рабочая версия нашего сайта». Спустя время появляется еще одна версия нашего сайта, допустим, 2.0. При возникновении проблем всегда можно откатиться до первой версии и запустить сайт заново.





## Контроль версий необходим, чтобы:

- \* хранить разные версии проекта;
- \* возвращаться к разным версиям проекта.

**Примеры:** версии сайтов (по датам) и документов.

Хранение версий сводится к созданию копий информации на компьютере или сервере. Функцию возврата реализуют за счёт восстановления предыдущих версий.

Таким образом, **система контроля** – это реализованная возможность замены информации с использованием сохраненных версий.



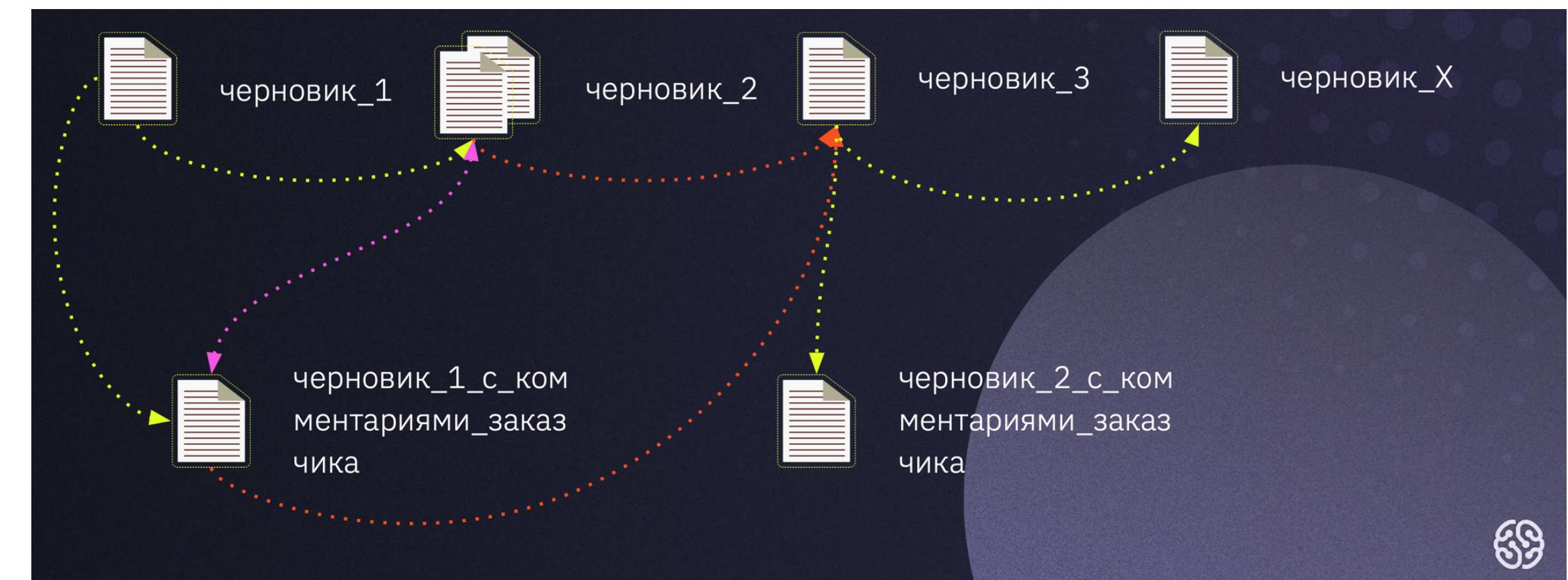
Когда вы работаете в команде, контроль версий помогает синхронизировать усилия.



## Примеры использования контроля версий в жизни:

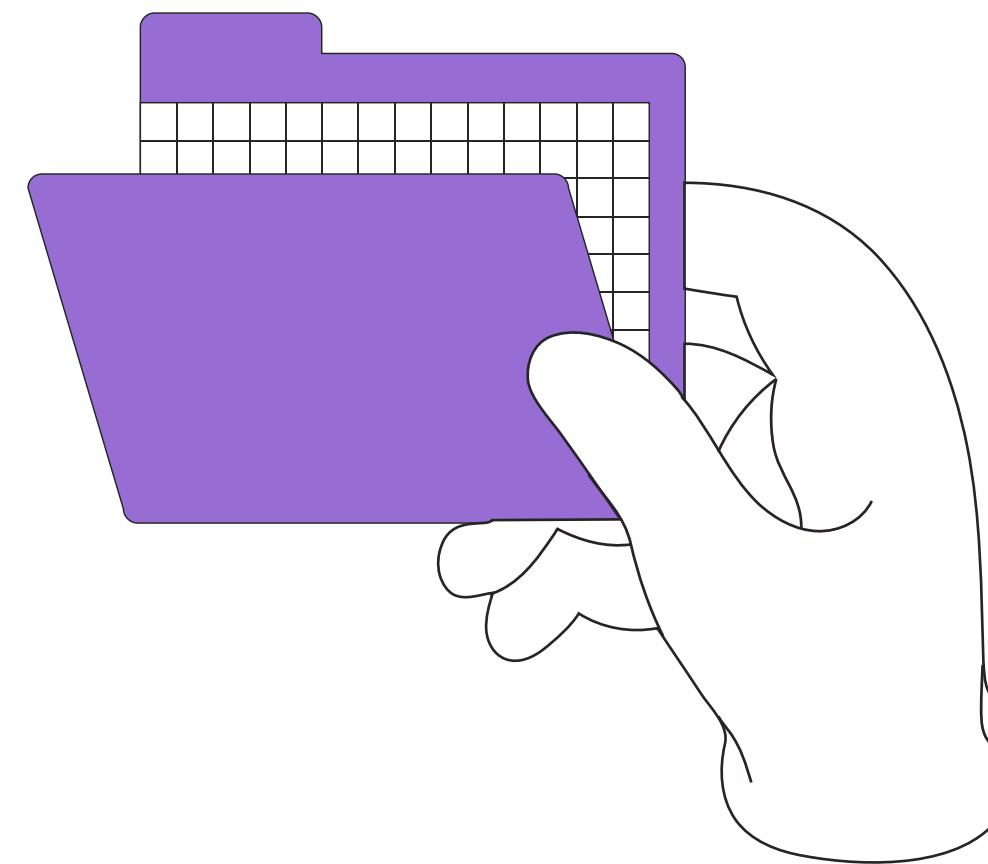
- \* Написание нескольких версий одного текста.
- \* Сохранение в компьютерных играх.
  - ✿ Если вы сделали что-то неправильно, вы всегда можете вернуться к состоянию, когда вас всё устраивало.
- \* Группа сотрудников пишет текст в течение нескольких дней.
  - ✿ Когда готов первый черновик, его отправляют на ревью коллеге. Пока он читает, другие сотрудники продолжают работу над текстом.

И когда ревьюер «выкапывает» комментарии к первой версии черновика, у команды уже появился «Черновик 2». После внесения правок появляется ещё и «Черновик 3». Файлы множатся и работа усложняется.



- ◆ Чтобы избежать описанной выше ситуации, команда обычно принимает правила именования и сортировки файлов. Но этого недостаточно для комфортной работы: всё равно приходится контролировать, когда пришёл документ, кто его владелец, каким образом документ был создан.

Можно организовать совместную работу с помощью сервиса Google Docs, но в реальных бизнес-ситуациях это не всегда удобно: чтобы контролировать изменения документа, его приходится править и перечитывать. А ещё работа в облачных сервисах — это угроза безопасности.

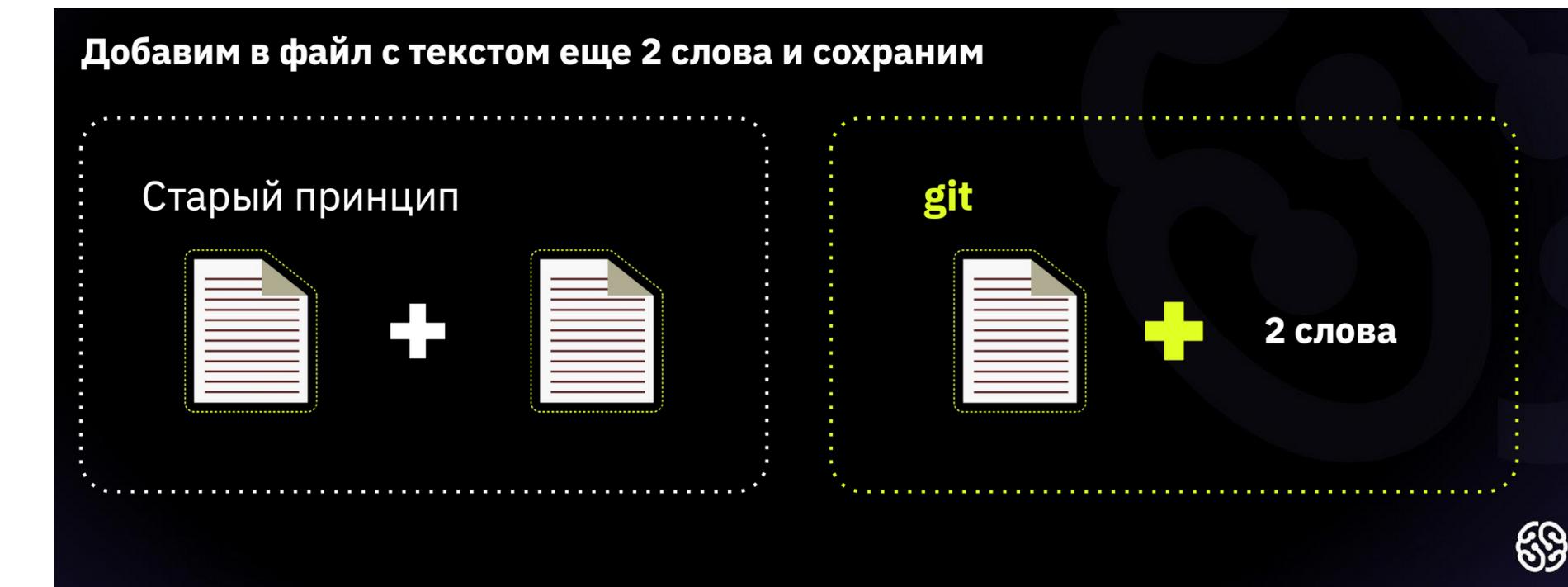


# Git - программа для контроля версий

В программировании проблемы совместной работы над проектами возникли ещё до появления облачных сервисов.

**!** Git – самая популярная система контроля версий, но не единственная. Алгоритм работы подобных систем схож.

Программа Git берёт на себя контроль версий проекта и позволяет переключаться между ними. Обратите внимание: Git хранит **не файлы целиком, а отличия между ними**. Это позволяет экономить память. Автор программы – Линус Торвальдс, создатель ОС Linux.



Для работы нужно установить Git: ссылку вы найдёте в своём личном кабинете на gb.ru, под видео лекции.

**Ваша задача на курс:** освоить систему контроля версий и с её помощью версионировать текст из файлов [readme.md](#), которые вы будете получать на семинарах.



## Команды Git

Осваивать Git проще процессе редактирования текстовых файлов. Markdown – язык разметки, который позволяет форматировать текст. Для написания в редакторе VS Code используется синтаксис языка.

Все команды задаём при помощи написания кода в терминале.

Прежде чем создавать репозиторий и инициализировать Git, проверим текущую установленную версию программы. Для этого в терминале введём команду:

```
git --version
```

Если Git установлен на компьютер, вы увидите его текущую версию.

Программа использует мнемонические команды, которые легко запомнить, если знать английский язык.



## Создание Git-репозитория:

- \* Берём локальный каталог, который не находится под версионным контролем, и превращаем его в репозиторий.
- \* Клонируем существующий репозиторий из любого места.

### Команда git init

- + Инициализация: указываем папку, в которой git начнёт отслеживать изменения
- + В папке создаётся скрытая папка .git

```
PS C:\Users\oleg-\OneDrive\Документы\testgit> git init
Initialized empty Git repository in C:/Users/oleg-/OneDrive/Документы/testgit/.git/
```

git init

### Команда git status

- + Показывает текущее состояние гита, есть ли изменения, которые нужно закоммитить (сохранить)

! Чтобы вызвать ранее введенную команду, пользуемся стрелками на клавиатуре. Перебираем недавно введенные команды нажатием стрелки «вверх».



## Команда git add

- + добавляет содержимое рабочего каталога в индекс (staging area) для последующего коммита. Эта команда дается после добавления файлов. Писать название целиком не обязательно: терминал дозаполнит данные автоматически.

PS C:\Users\oleg-\OneDrive\Документы\Lection1Prepare> git add first.txt ← Расширение файла  
PS C:\Users\oleg-\OneDrive\Документы\Lection1Prepare> [ ] ↑ [ ] ↑ [ ] ↑ ПРОБЕЛ ПРОБЕЛ Название файла. Расширение

The screenshot shows a terminal window with the following text:  
PS C:\Users\oleg-\OneDrive\Документы\Lection1Prepare> git add first.txt ← Расширение файла  
PS C:\Users\oleg-\OneDrive\Документы\Lection1Prepare> [ ] ↑ [ ] ↑ [ ] ↑ ПРОБЕЛ ПРОБЕЛ Название файла. Расширение

Annotations in yellow and pink highlight specific parts of the command:

- A yellow box surrounds the file name "first.txt". A yellow arrow points from the text "Расширение файла" (File extension) to this box.
- Pink arrows point from the text "ПРОБЕЛ" (Space) to the two spaces between "git add" and the file name.
- A pink bracket below the third space is labeled "Название файла. Расширение" (File name. Extension).

git add



## Команда **git commit**

- + зафиксировать или сохранить

По умолчанию **git commit** использует лишь этот индекс, так что вы можете использовать **git add** для сборки слепка вашего следующего коммита.

Команда **git commit** берёт все данные, добавленные в индекс с помощью **git add**, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

```
PS C:\Users\oleg-\OneDrive\Документы\Lection1Prepare> git commit -a -m "First.txt changed"
[master ea7c313] First.txt changed
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Кавычки обязательны с двух сторон

ПРОБЕЛ

ПРОБЕЛ

Флаг сообщения

Текст сообщения

git commit



## Команда git log

- + Журнал изменений
- + Перед переключением версии файла в Git используйте команду `git log`, чтобы увидеть количество сохранений

```
PS C:\Users\oleg-\OneDrive\Документы\Lecture1Prepare> git log
commit 6f39027914e34f40f432a63aba6364938640ecc9 (HEAD ->master)
Author: oleg golенищев <oleg-golenishev63@yandex.ru>
Date:   Thu Sep 30 13:01:28 2021 +0300

    added list

commit 3432a6393ccb49cccd7434b04adc9e0ee1f607068
Author: oleg golенищев <oleg-golenishev63@yandex.ru>
Date:   Thu Sep 30 12:56:56 2021 +0300

    changed readme

commit 95b35e1f5ac4f9c971abed2813212e7d9165726b
Author: oleg golенищев <oleg-golenishev63@yandex.ru>
Date:   Thu Sep 30 12:31:04 2021 +0300

    first commit
```

ПРОБЕЛ  
ОБЯЗАТЕЛЬНО

Номер  
коммита

Сообщение к коммиту

git log

## Команда git checkout

- + Переключение между версиями.
- + Для работы нужно указать не только интересующий вас коммит, но и вернуться в тот, где работаем, при помощи команды `git checkout master`.



## Команда git diff

- + Показывает разницу между текущим файлом и сохранённым
- + Перед переключением версии файла в Git используйте команду git log, чтобы увидеть количество сохранений

```
PS C:\Users\oleg-\OneDrive\Документы\Lecture1Prepare> git diff
diff --git a/Readme.md b/Readme.md
index 5274727..c1f0625 100644
--- a/Readme.md
+++ b/Readme.md
@@ -6,4 +6,8 @@
# Ещё их можно обозначать вот так
-## a можно и так
\ No newline at end of file } Измененная строка
+## a можно и так
+
+добавим список
+* Первый элемент
....skipping...
diff --git a/Readme.md b/Readme.md
index 5274727..c1f0625 100644
--- a/Readme.md
+++ b/Readme.md
@@ -6,4 +6,8 @@
```

git log

## Синтаксис языка Markdown.

- + Жирный текст – \*
- + Курсивный текст – \*
- + Зачеркнутый текст – ~
- + Выделяют заголовки – # в начале строки
- + Показать уровень заголовка – подчеркивание знаками = или \*\*\*
- + Нумерованные Списки – обозначаются обычными цифрами 1, 2, 3
- + Ненумерованные Списки – **обозначаются \*знаками в начале строки**
- + Вложенные Списки – выполняем отступы



## Заголовки на языке маркдаун на гитхаб.



Git отслеживает файлы по имени!

Если изменить имя файла, необходимо добавить файл с новым именем + git commit

