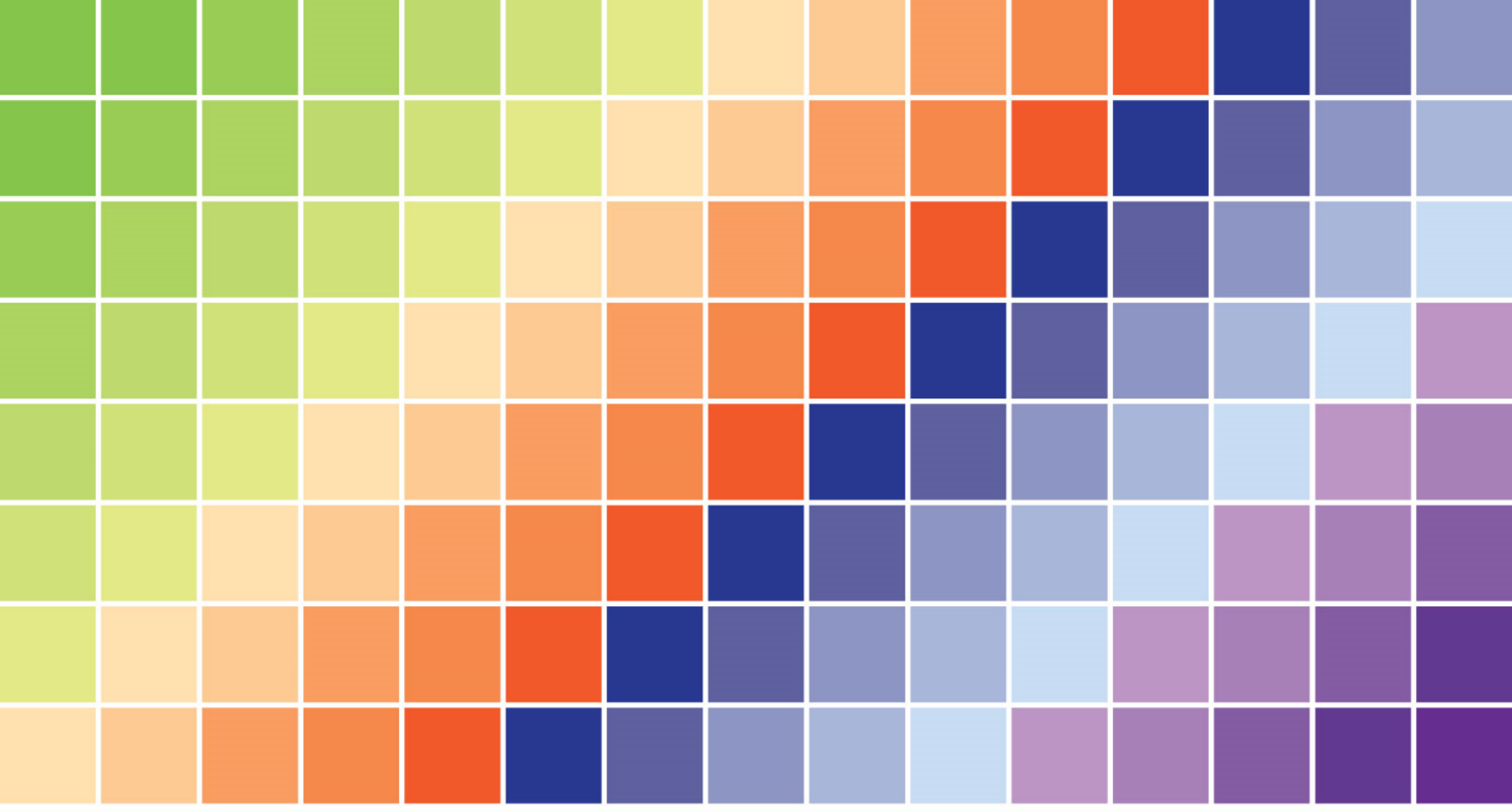


School of Computer Science

СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ. GIT

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

Как хранить исходные тексты?

Существование без систем контроля версий

- Нет резервных копий
- Копии файла
  - lab.cpp
  - lab_new.cpp
  - lab_newnew.cpp
  - lab_final.cpp
  - lab_final_v2.cpp
  - ...
- Копии папки/архива

Свойства системы контроля версий

Система контроля версий

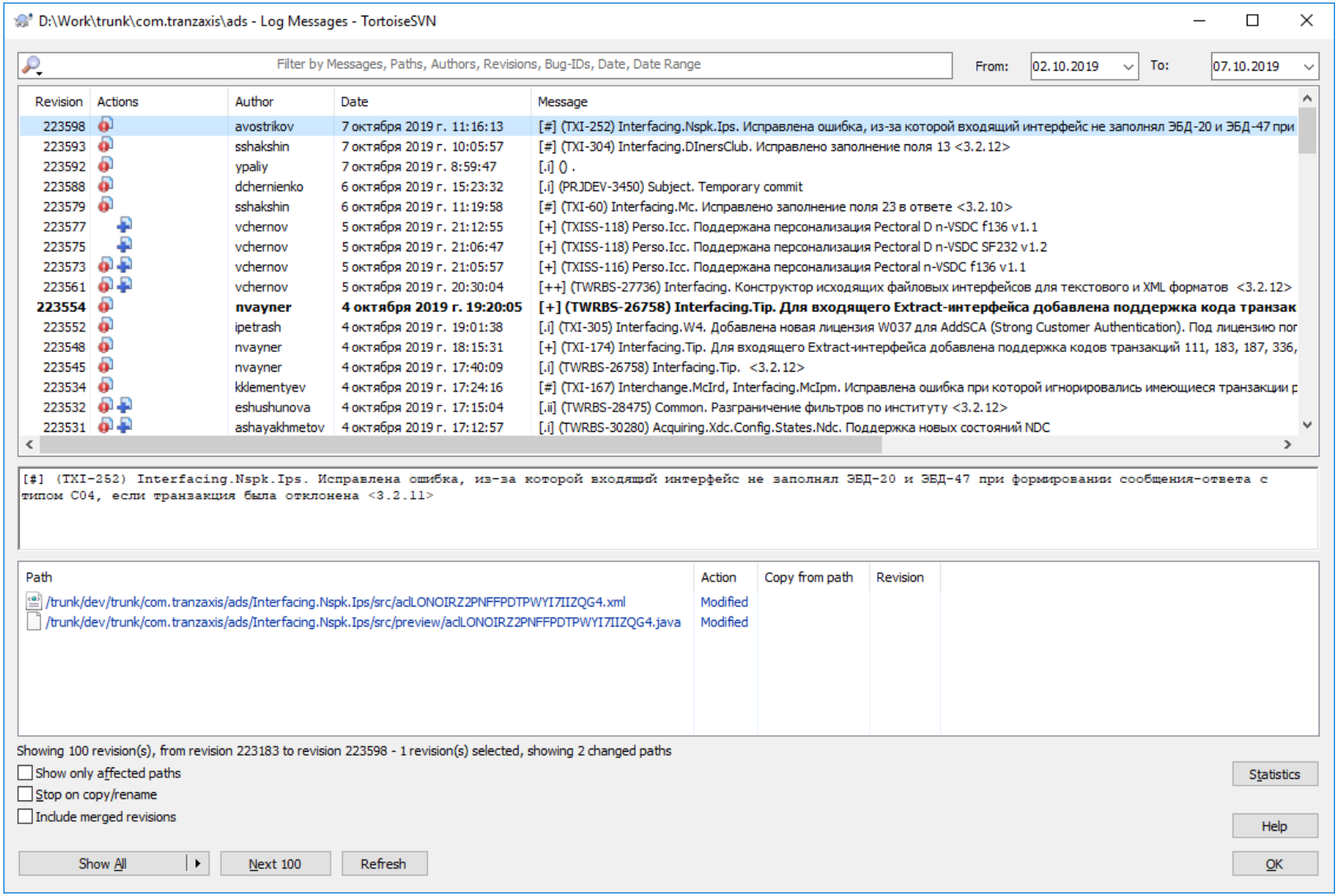
- Выглядит как одна копия всего проекта
- Хранит все версии, начиная с самой 1-ой
- Есть возможность групповой работы
- История изменений для каждого файла
- Просмотр изменений между двумя версиями файла

Различные системы контроля версий

Системы контроля версий

- RCS
- CVS
- Subversion (SVN)
- Git
- Mercurial
- ...

История изменений в системе контроля версий



ОПРЕДЕЛЕНИЕ И КЛАССИФИКАЦИЯ

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов

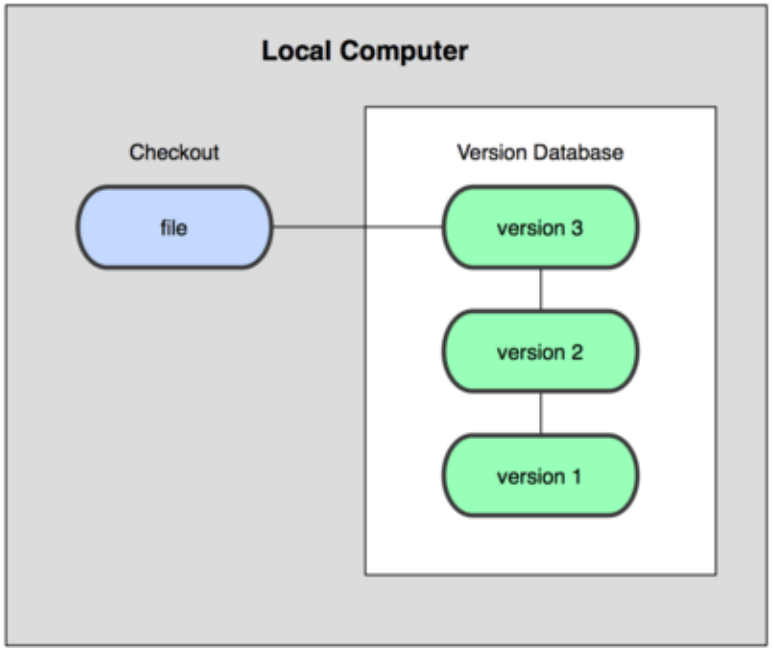
ЛОКАЛЬНЫЕ СКВ

Суть: локальная база данных, в которой хранится список изменений исходного файла

Особенности:

- Работа только с одним файлом;
- Невозможность одновременной работы нескольких пользователей с системой;
- Риск потери данных.

Представители: RCS (Revision Control System)

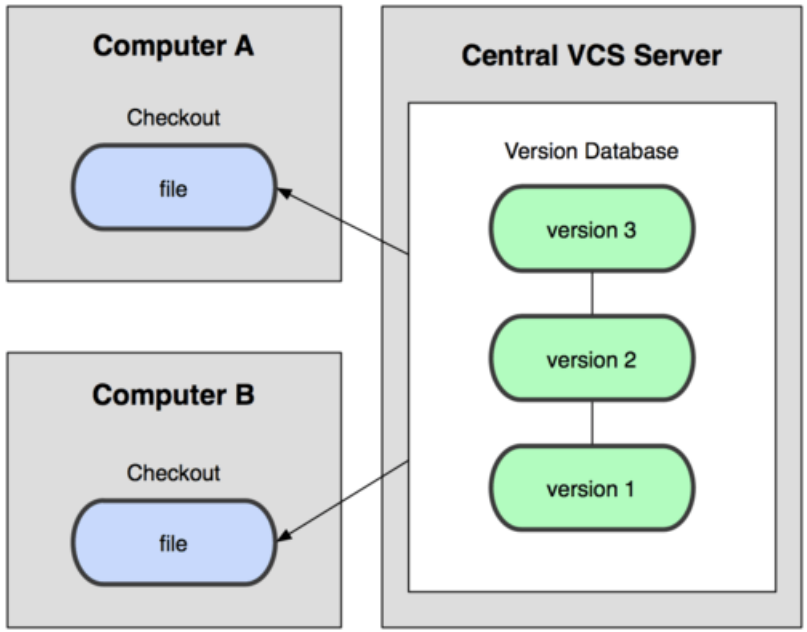


ЦЕНТРАЛИЗОВАННЫЕ СКВ

Суть: центральный сервер, на котором хранятся все файлы под версионным контролем; клиенты получают копии файлов

Плюсы:

- Под контролем находятся все файлы;
- Возможность работы нескольких пользователей;
- Удобство администрирования.



Минусы:

- Риск потери данных.

Представители: CVS, SVN

В Компас Плюс используется для работы Subversion (SVN)

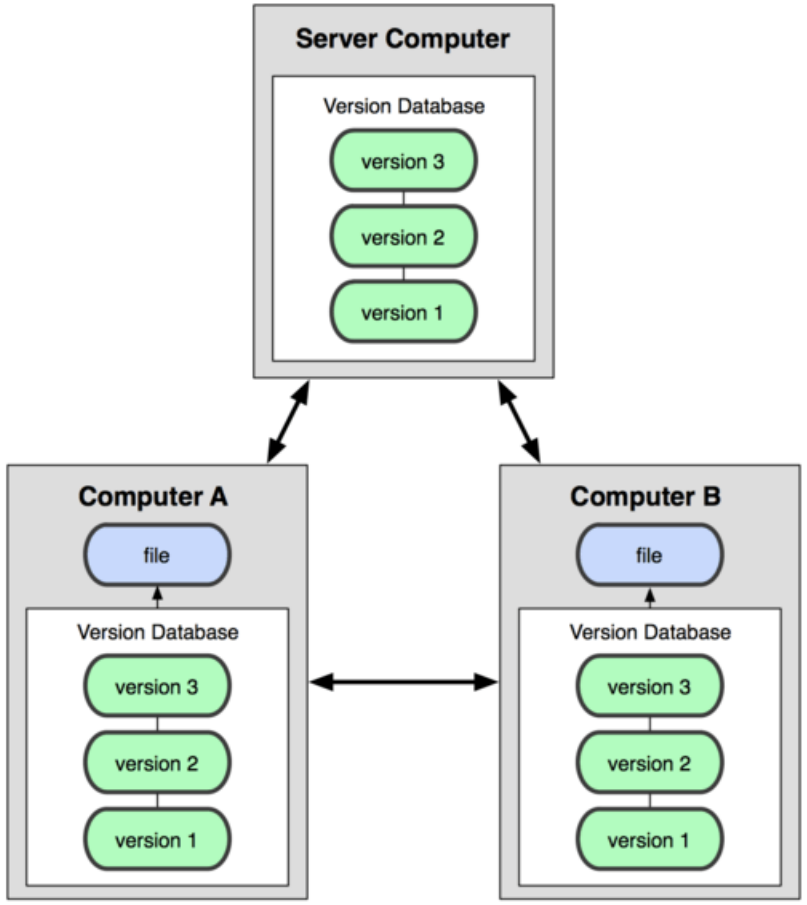
РАСПРЕДЕЛЕННЫЕ СКВ

Суть: клиенты забирают весь репозиторий, который в любой момент может быть скопирован обратно на сервер

Особенности:

- Отсутствие четко выделенного центрального хранилища версий – *репозитория*
- Возможность работы с несколькими удалёнными репозиториями

Представители: Git, Mercurial



В Компас Плюс используется для работы Git

ЧТО ТАКОЕ GIT

Git – распределенная система контроля версий

Разработан: Линусом Торвальдсом в 2005 году

Проекты: ядро Linux, Swift, Android, jQuery, PHP, Qt, ряд дистрибутивов Linux

Сервисы для работы: GitHub, GitLab, BitBucket, TortoiseGit

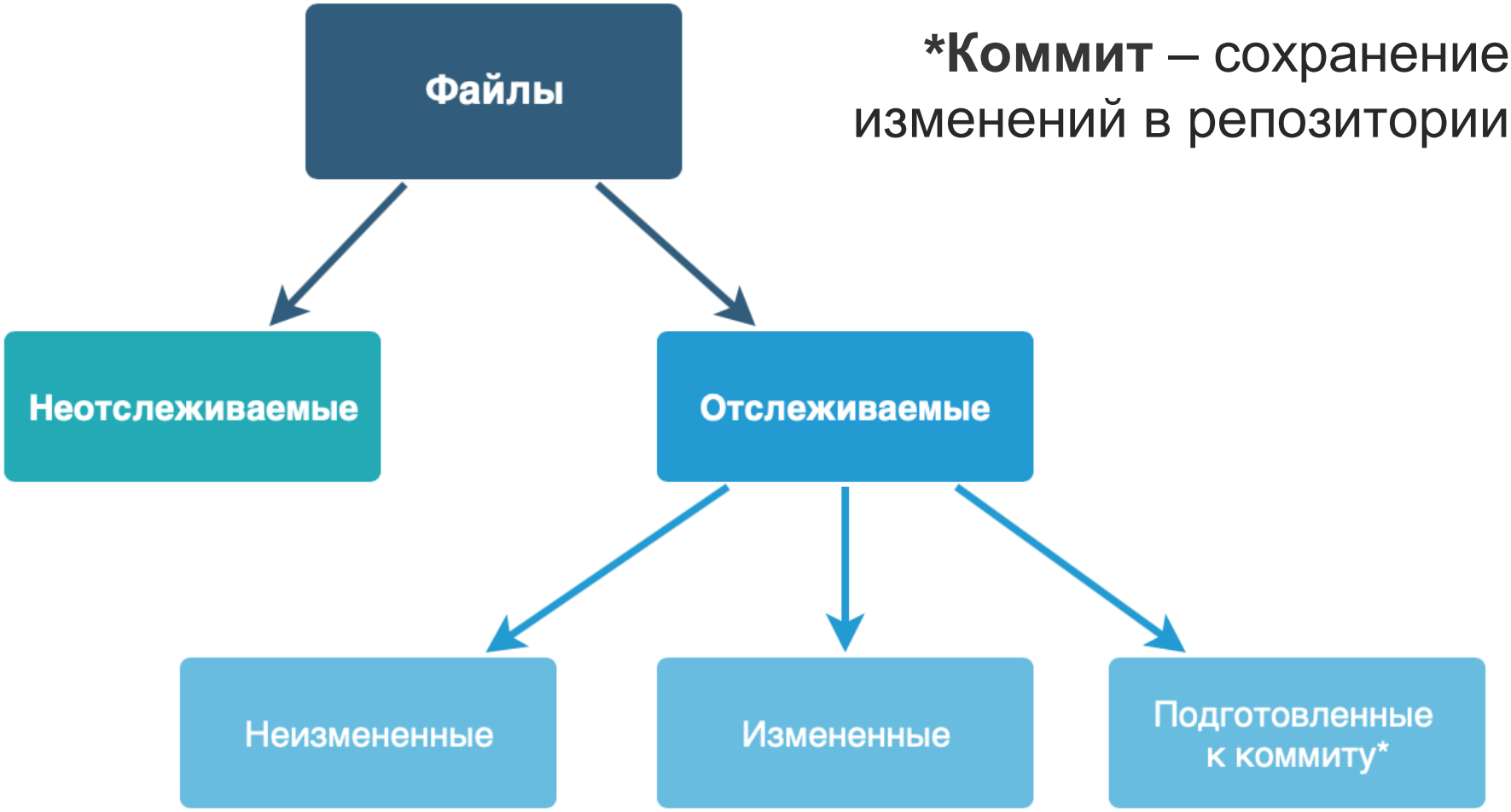
РЕПОЗИТОРИИ

Репозиторий – место, где хранятся и поддерживаются какие-либо данные:

- **Локальный** – место на вашем компьютере
- **Удаленный** – хранилище в сети (например, GitHub)

Удаленный репозиторий принято называть **origin**

СОСТОЯНИЯ ФАЙЛОВ В РАБОЧЕМ КАТАЛОГЕ



***Коммит** – сохранение изменений в репозитории

Уже зафиксированные в коммите и сохраненные в локальном репозитории (.git)

Файлы, которые поменялись, но еще не зафиксированы

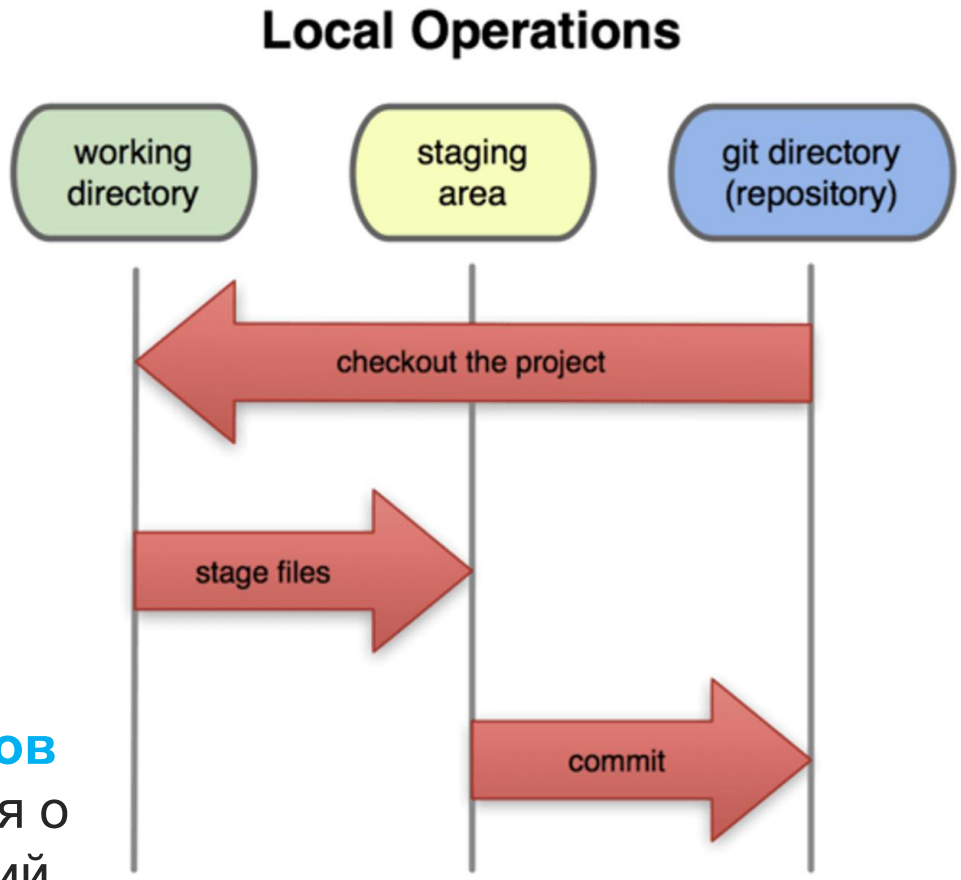
Измененные и подготовленные для коммита – staged for commit (проиндексированные)

КАТАЛОГИ GIT

Каталог Git – место, где Git хранит метаданные и базу данных (репозиторий) проекта, именуется “**.git**” – это скрытый каталог

Рабочий каталог – копия определенной версии проекта, которая время от времени синхронизируется с удаленным сервером и **содержит в себе Каталог Git**

Область подготовленных файлов (индекс) – служебная информация о том, что должно войти в следующий коммит



ВАРИАНТЫ РАБОТЫ С GIT

Через командную строку (CLI) и графический интерфейс (GUI)

НАЧАЛО РАБОТЫ

- Установить git:

  https://git-scm.com/downloads
- Перейти в командную строку
- Указать данные пользователя:

```
$ git config --global user.name ”UserName”
```

```
$ git config --global user.email myname@example.com
```

- Перейти в каталог проекта и создать в нем git-репозиторий:

```
$ git init
```

ОСНОВНЫЕ КОМАНДЫ

- Текущее состояние репозитория:

```
$ git status
```

- Добавить файл под версионный контроль/к коммиту

(другое название – «*Проиндексировать изменения*»):

```
$ git add <filename>
```

- Коммит (только для проиндексированных файлов):

```
$ git commit –m ”commit message”
```

- Commit message должен быть осмысленным и четко передавать содержание коммита
- Обычно придерживаются принципа «Один завершенный цикл разработки – один коммит»

ОСНОВНЫЕ КОМАНДЫ
Удаление файла: git rm

- Удаление файла из Git и рабочего каталога:

```
$ git rm <filename>
```

- Удаление файла из Git, оставив его в рабочем каталоге:

```
$ git rm --cached <filename>
```

```
slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ ls
file1  file2  file3

slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ git rm file2
rm 'file2'

slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ git rm --cached file3
rm 'file3'

slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ ls
file1  file3

slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file2
        deleted:    file3

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file3
```

ОСНОВНЫЕ КОМАНДЫ
Просмотр истории коммитов: git log

- Просмотр всей истории коммитов в обратном хронологическом порядке:

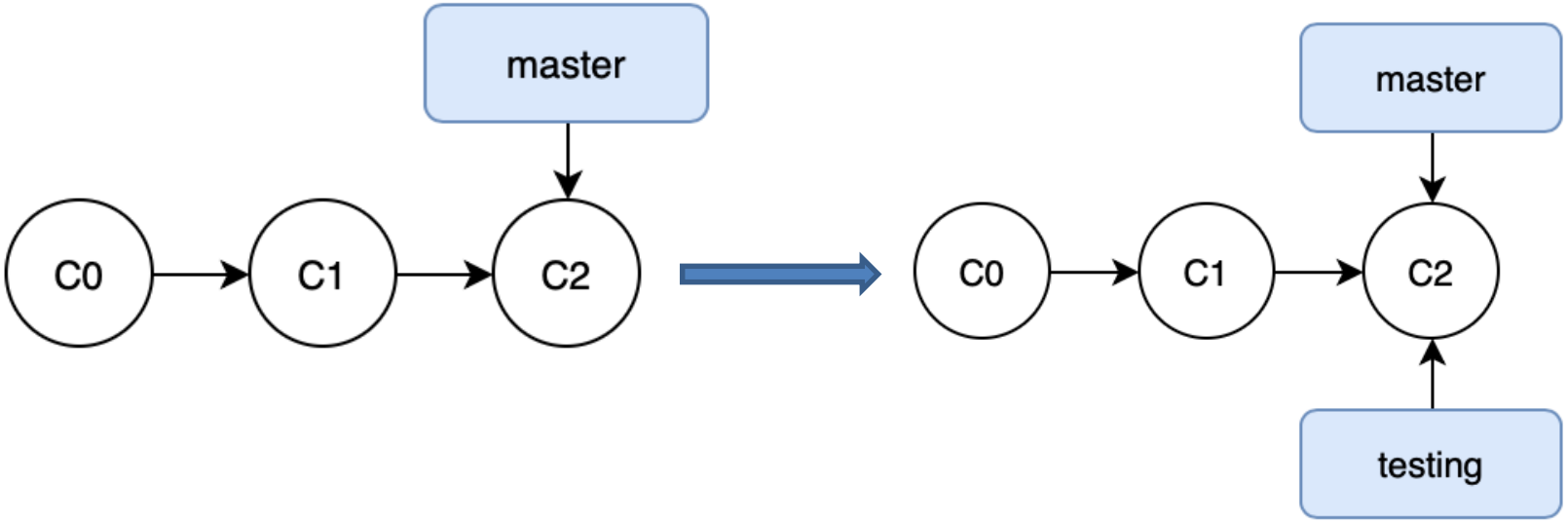
```
$ git log
```

```
slukashenko@SLUKASHENKO-WIN MINGW64 /c/test_git (main)
$ git log
commit 4e2152ed6617ae79d16a573f9f97017401b1cc0f (HEAD -> main)
Author: slukashenko <slukashenko75@gmail.com>
Date:   Sat Sep 11 14:51:06 2021 +0500

    Added 3 files

commit 42b03725088eff7474605580ad6e29f7563c61f5
Author: slukashenko <slukashenko75@gmail.com>
Date:   Sat Sep 11 14:49:12 2021 +0500

    Deleted text.txt

commit ce5fe7cff9a641e43b57ef7a28d3c9846c832c68
Author: slukashenko <slukashenko75@gmail.com>
Date:   Sat Sep 11 14:35:53 2021 +0500

    Added 1st text file
```

Ветка (branch) – это подвижный указатель на один из коммитов. Ветка по умолчанию называется *master (main, trunk, …)*, т.к. *master* посчитали неполиткорректным ☺

Создание новой ветки:

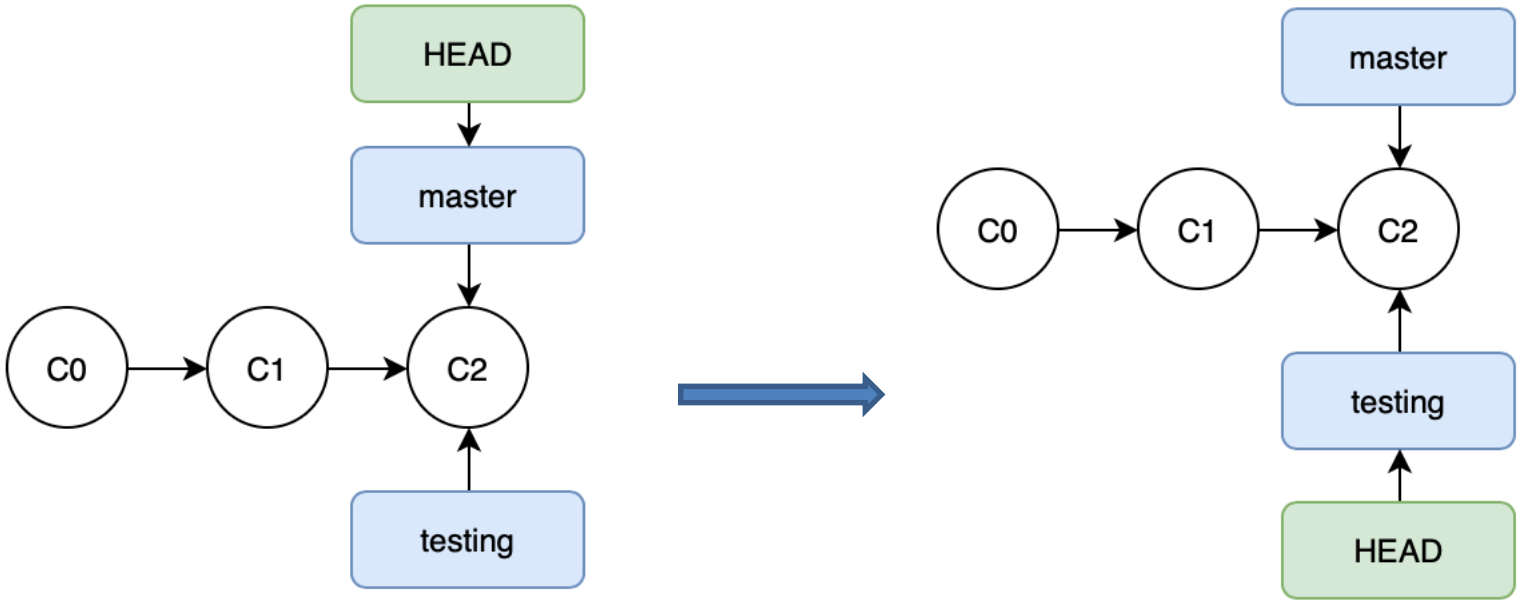
```
$ git branch testing
```



HEAD – это указатель на текущую ветку

Перейти на другую ветку:

```
$ git checkout testing
```



СЛИЯНИЕ

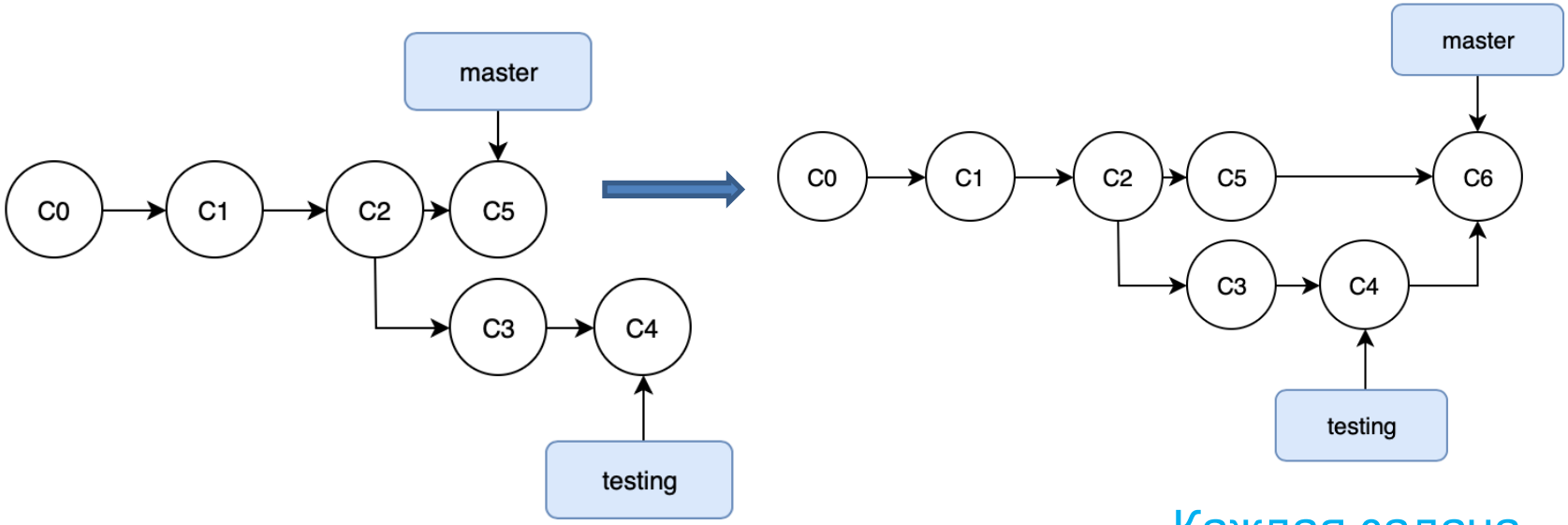
Возврат на основную ветку:

```
$ git checkout master
```

Слияние веток:

```
$ git merge testing
```

При слиянии возможны конфликты!



Удаление ветки:

```
$ git branch -d testing
```

Каждая задача решается в своей ветке

ЦЕНТРАЛЬНЫЙ РЕПОЗИТОРИЙ
Например, на хостинге GitHub или GitLab

- Связать центральный репозиторий с удаленного хоста для синхронизации с локальным репозиторием:

```
$ git remote add origin <url>
```

- Клонировать центральный репозиторий с удаленного хоста на свой компьютер:

```
$ git clone <url> [local_directory]
```

```
slukashenko@SLUKASHENKO-WIN MINGW64 /c
$ git clone https://github.com/ITFI-school/python-course-2021-2022-11.git course_dir
Cloning into 'course_dir'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15 (delta 2), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), 1.39 MiB | 3.54 MiB/s, done.
Resolving deltas: 100% (2/2), done.

slukashenko@SLUKASHENKO-WIN MINGW64 /c
$ cd course_dir

slukashenko@SLUKASHENKO-WIN MINGW64 /c/course_dir (main)
$ ls -a
 ./   ../   .git/   .gitignore  'Lesson 1'/   README.md

slukashenko@SLUKASHENKO-WIN MINGW64 /c/course_dir (main)
```

ЦЕНТРАЛЬНЫЙ РЕПОЗИТОРИЙ
Извлечение данных

- Только получение данных:

```
$ git fetch origin [ветка]
```

- Получение и слияние данных:

```
$ git pull origin [ветка]
```

Пример:

```
$ git pull origin main
```

```
slukashenko@SLUKASHENKO-WIN MINGW64 /c/course_dir (main)
$ git pull origin main
From https://github.com/ITFI-school/python-course-2021-2022-11
 * branch            main       -> FETCH_HEAD
Already up to date.
```

ЦЕНТРАЛЬНЫЙ РЕПОЗИТОРИЙ
Отправка данных

- Отправка данные на удаленный хост:

```
$ git push origin [ветка]
```

```
$ git push origin main
```

Выполняется успешно если:

1. Есть права на запись в центральный (удаленный) репозиторий
2. У вас находится актуальная версия репозитория (никто не делал *git push* с момента вашей последней синхронизации: *git pull origin main*)

GITHUB И GITLAB

GitHub и **GitLab** – сервисы для онлайн-хостинга проектов, использующих Git
Цель создания: содействовать взаимодействию разработчиков

Функции:

- Разработка open-source проектов
- Контроль доступа
- Багтрекинг
- Управлением задачами
- Документация к проекту

Страничка на GitHub – лицо разработчика

ПОЛЕЗНЫЕ ССЫЛКИ

Книга “Pro Git” с переводом на русский:
https://git-scm.com/book/ru/v2

Интерактивные курсы по использованию Git:
https://githowto.com, https://practicum.yandex.ru/git-basics

Шпаргалки по командам Git:
https://eax.me/git-commands/
https://training.github.com/downloads/ru/github-git-cheat-sheet/

Статья ”Git – советы новичкам”:
https://habr.com/en/company/playrix/blog/345732/

Видео на YouTube:

1. *Git и GitHub Курс Для Новичков*
2. *Git. Большой практический выпуск*
3. *Про системы контроля версий из курса МФТИ "Практика программирования на Python"*

GitHub Desktop

-GitHub Desktop — это бесплатное приложение, которое помогает работать с файлами, размещенными на GitHub.

-Особенностью Desktop является наличие графического интерфейса

-После первого входа в GitHub Desktop вас попросят ввести ваши логин и пароль от GitHub.com. После этого у вас появится доступ ко всем репозиториям, сохранённым в профиле.

Клонирование репозитория

1. Откройте GitHub Desktop на вашем компьютере.

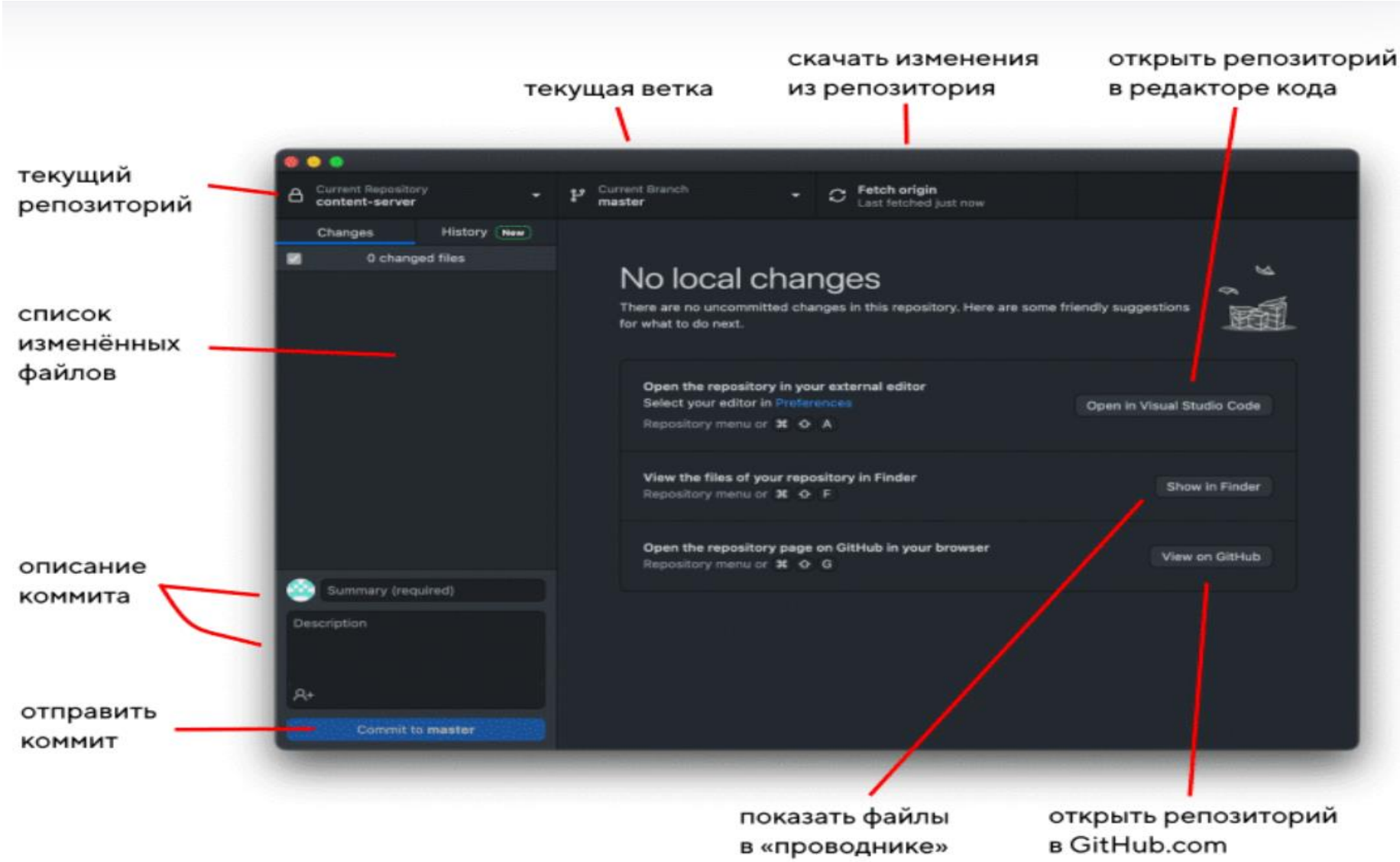
2. Нажмите кнопку “File” в верхней панели меню и выберите “Clone Repository” из выпадающего списка.

3. В появившемся окне введите URL-адрес репозитория, который вы хотите склонировать. URL-адрес можно найти на странице репозитория в GitHub.

4.Выберите путь, по которому хотите сохранить локальную копию репозитория на вашем компьютере.

5.Нажмите кнопку “Clone” чтобы начать клонирование репозитория.

Рабочая область GitHub Desktop



ПОЛЕЗНЫЕ ССЫЛКИ

Подробная настройка и установка GitHub Desktop:
Github для самых маленьких #2 | Практика – GitHub Desktop | Приложение Гитхаб для компьютера - YouTube
Установка GitHub Desktop:
https://desktop.github.com/
Полная документация GitHub Desktop от GitHub:
https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science