

# Работаем с системой контроля версий

HTML course: Lesson 3



# План **урока**

1

Как происходит процесс разработки

2

Что такое система контроля версий и почему их используют

3

Инструменты для работы с git

4

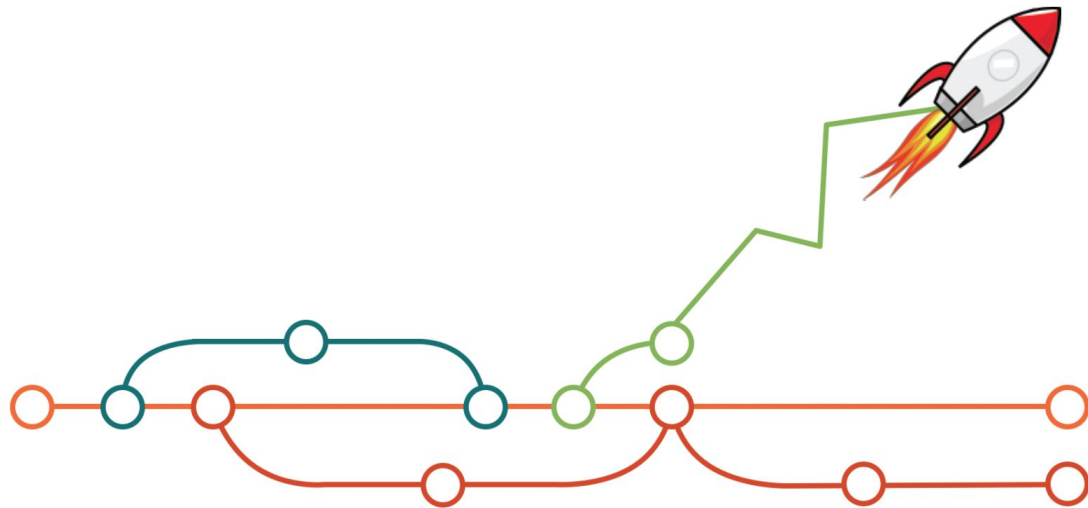
Основные команды для работы

# Команда

- Project Manager – управляет сроками и взаимодействует с другими командами
- Product Owner – управляет планами и пишет требования
- UI/UX designer – создает визуальную интерфейс
- Team Lead – отвечает полностью за
- Markup developer – верстает сайт по макету
- Frontend developer – пишет функционал (оживляет сайт)
- QA – ищет ошибки



# Работа в команде



# Система контроля версий

## Version Control System

– это система для управления версиями исходного кода программ.

**Git** – это распределённая и децентрализованная система управления версиями файлов

Mercurial

Bazaar

# Зачем использовать **Git** (самая популярная система контроля версий)

- Хранение полной истории изменений
- Описание причин всех сделанных изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок
- Совместная работа команды над одним проектом
- Возможность изменять код, не мешая работе коллег



# Сервисы для работы с GIT



GitHub.com

- социальная сеть
- git хранилище репозиториев
- место хранения open source проектов
  - ◆ можно посмотреть как работает чужой код
  - ◆ использовать его в своем проекте
  - ◆ предложить свои изменения
- место для хранения портфолио с возможностью опубликовать свой проект
- место для резюме




GitLab.com



BitBucket.org

# Способы работы с git

- Console / Terminal
- GUI (graphical user interface)
  - SourceTree
  - GitKraken
  - GitHub Desktop
- Source-code editor
  - VS Code 
  - IntelliJ IDEA
  - Atom





# Репозиторий

– хранилище проекта и его истории

- файлы конфигураций (настроек)
- файлы журналов операций (история изменений)
- индекс расположения файлов (их место в папках проекта)
- сами файлы проекта

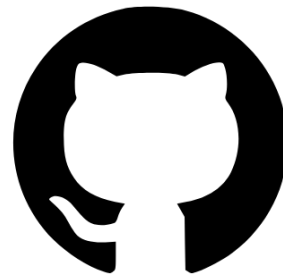
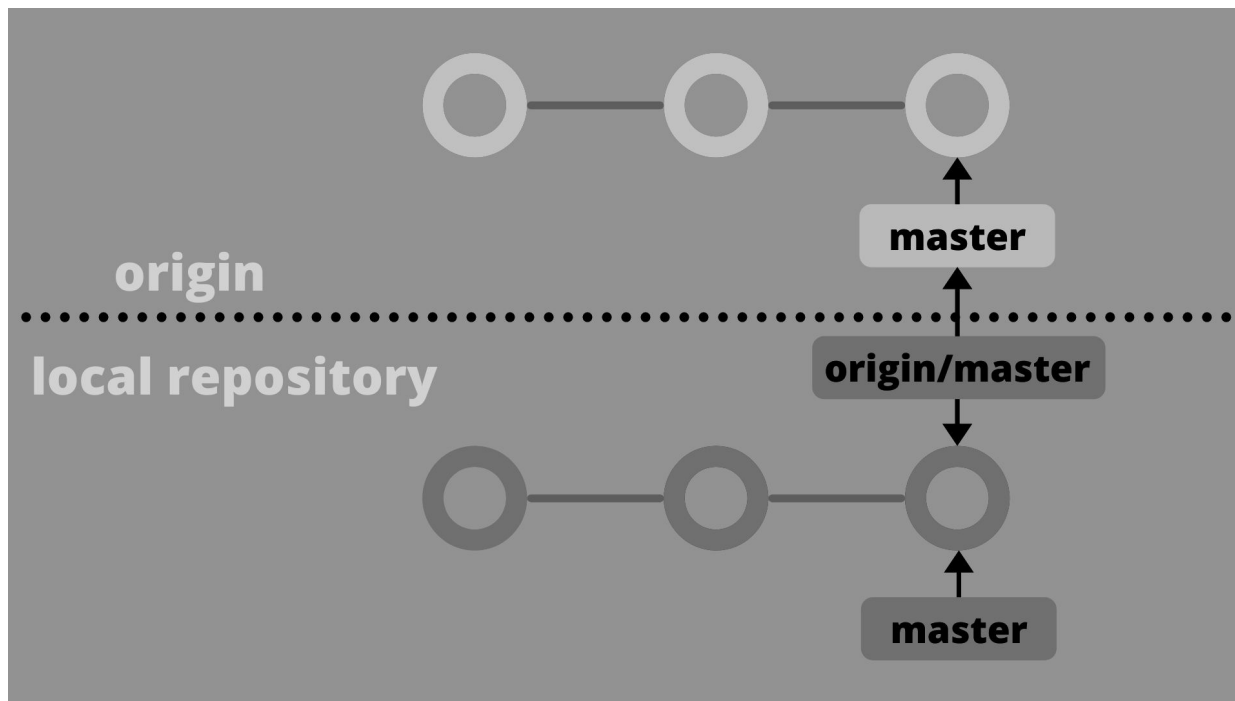


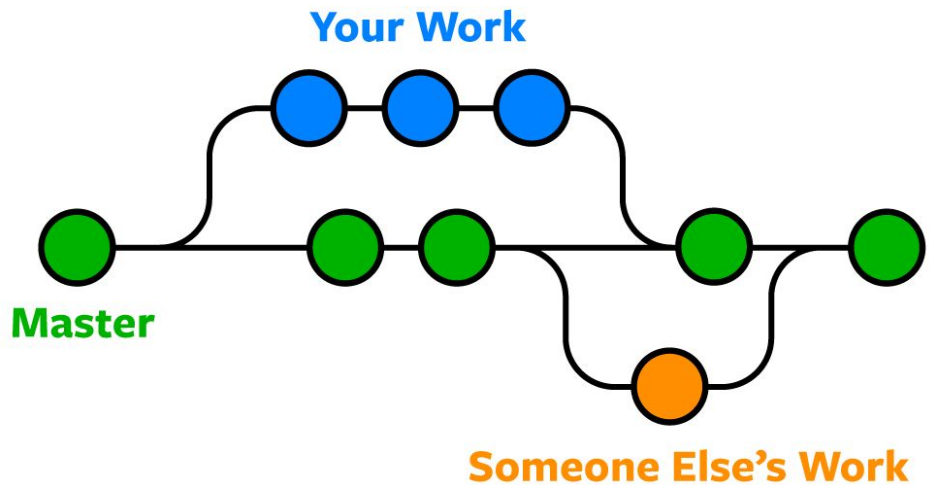
**Локальный репозиторий** – репозиторий с которым идет работа на компьютере.

**Удаленный репозиторий** – тот, что размещен на удаленном сервере. Тут собираются все изменения, внесенные в проект, и здесь же их можно взять при необходимости.

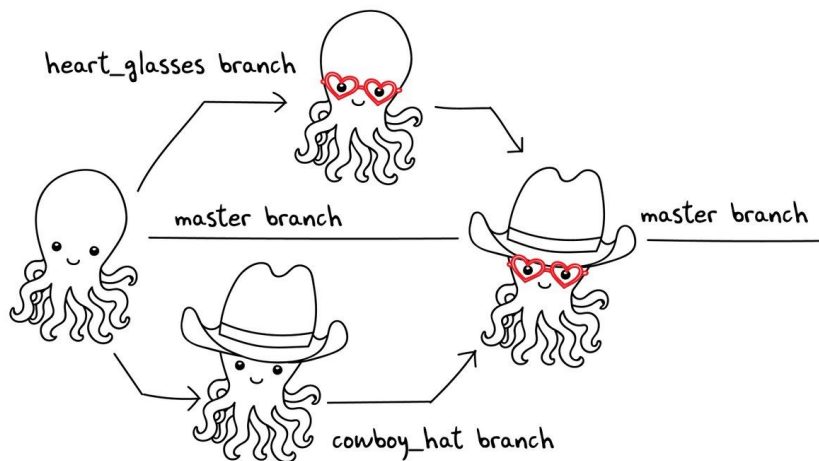
# git origin

– псевдоним удаленного репозитория, который находится на github (его можно поменять).





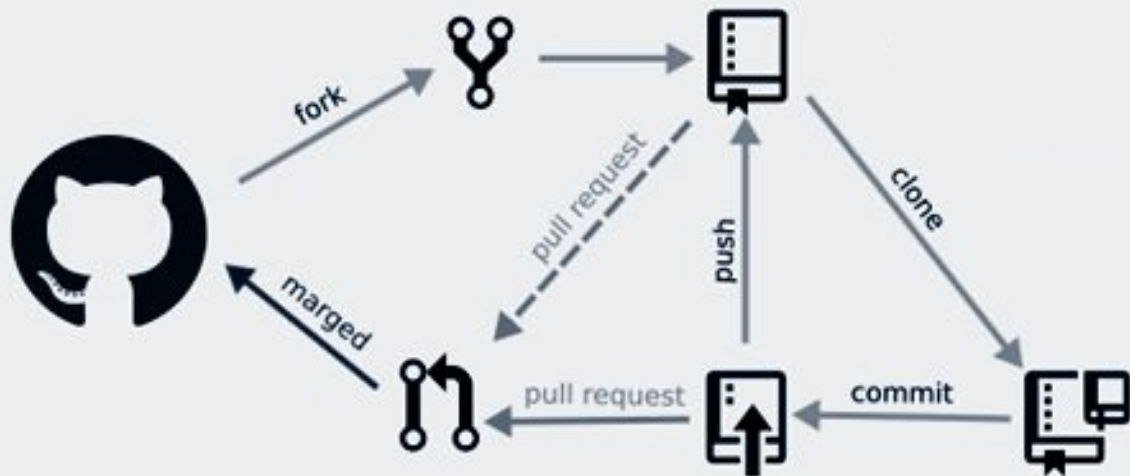
# git branch



**Master** – главная ветка любого репозитория.

Остальные ветки можно называть как угодно.

# Процесс работы с репозиторием



# Термины **GIT**

## **Fork**

— копия репозитория.

## **Clone**

— копирование репозитория на свой компьютер.

## **Commit**

— фиксация изменений.

## **Branch**

— ветка, параллельное «ответвление» в репозитории.

## **Pull**

— скачивание последних сохраненных изменений с удаленного репозитория.

## **Push**

— отправка свежих коммитов в удаленный репозиторий.

# Термины **GIT**

## **Pull Request**

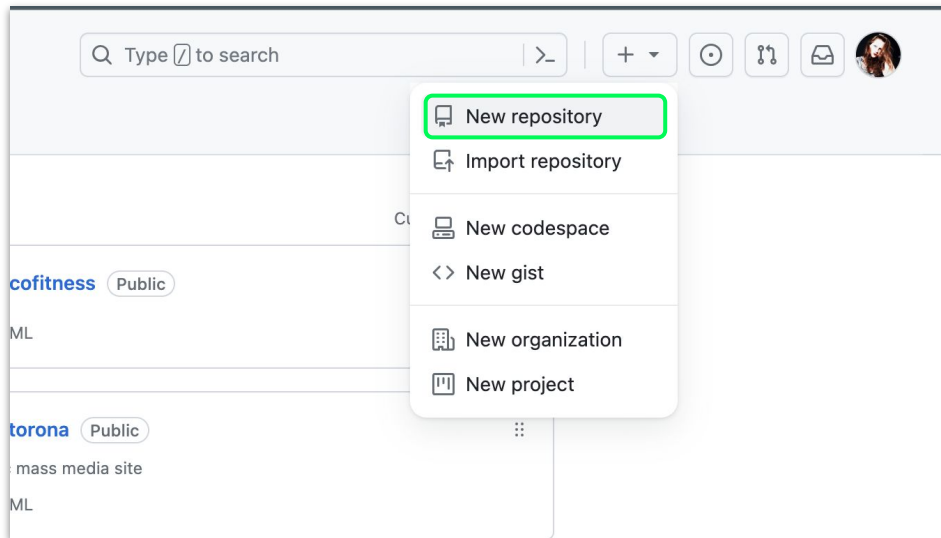
- запрос на объединение основного репозитория с его форком (или на объединение веток).

## **Merge**

- слияние изменений, произведенных в его ветке или форке.

## **Code Review**

- проверка кода (по внешнему виду, способности решать поставленные задачи, по соответствию требованиям).

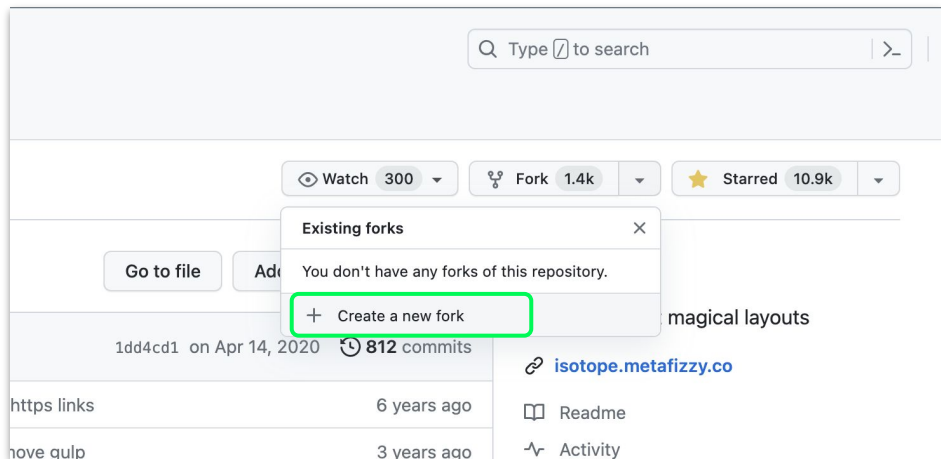


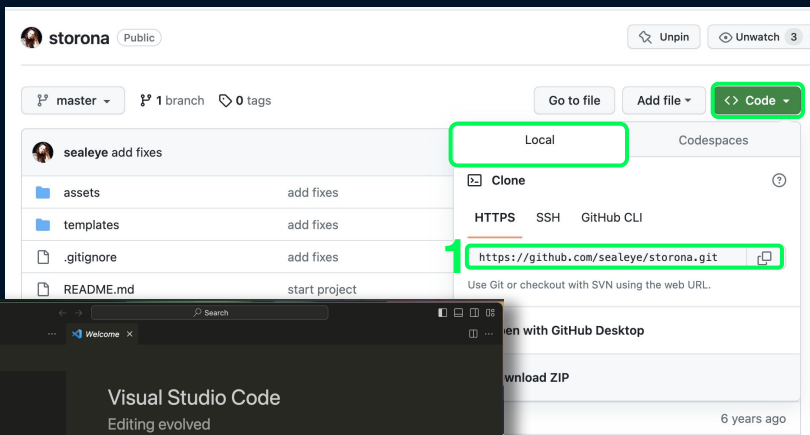
1

# Создаем репозиторий

или

делаем **Fork** уже существующего репозитория

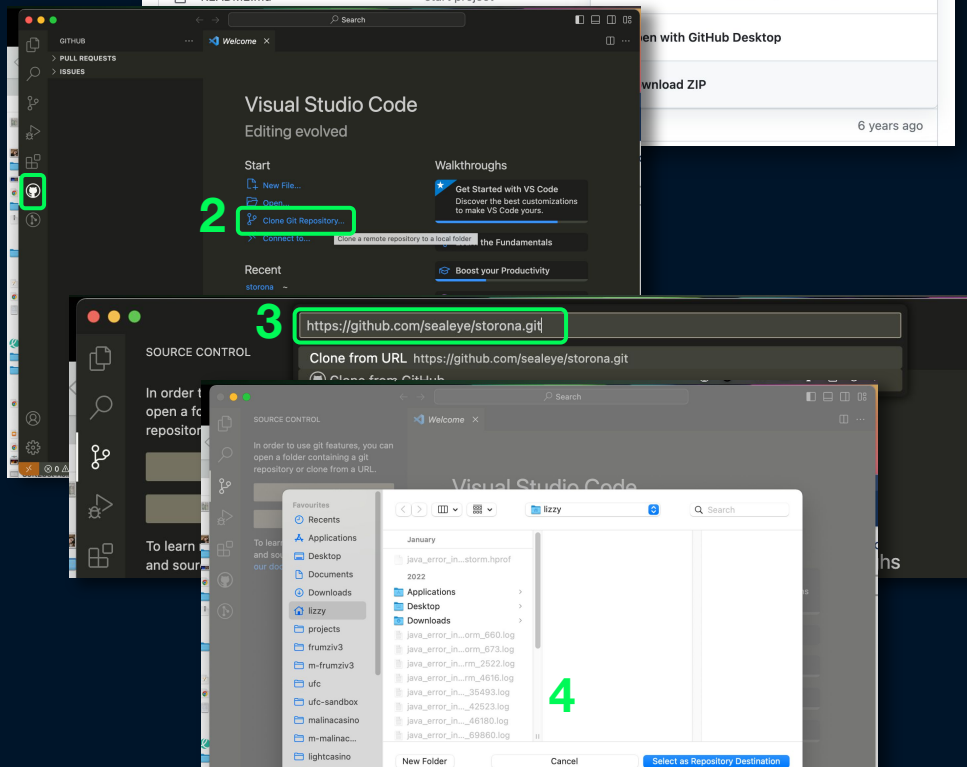




## 2 Clone

Используем VS Code

1. **Копируем** ссылку на репозиторий
2. На главной странице редактора кода выбираем пункт **Clone Git Repository**
3. **Вставляем** ссылку в верхнюю строку
4. **Выбираем папку**, где на компьютере будет лежать папка с кодом



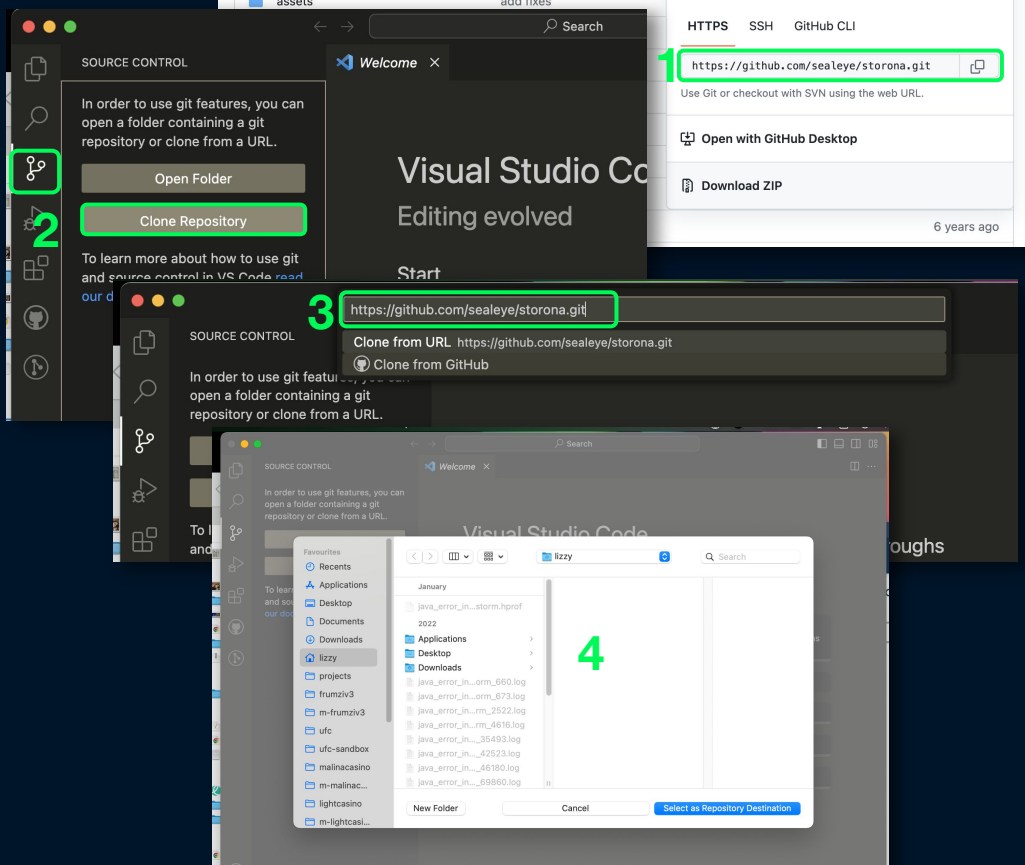


## 2.2

# Clone

## Используем VS Code

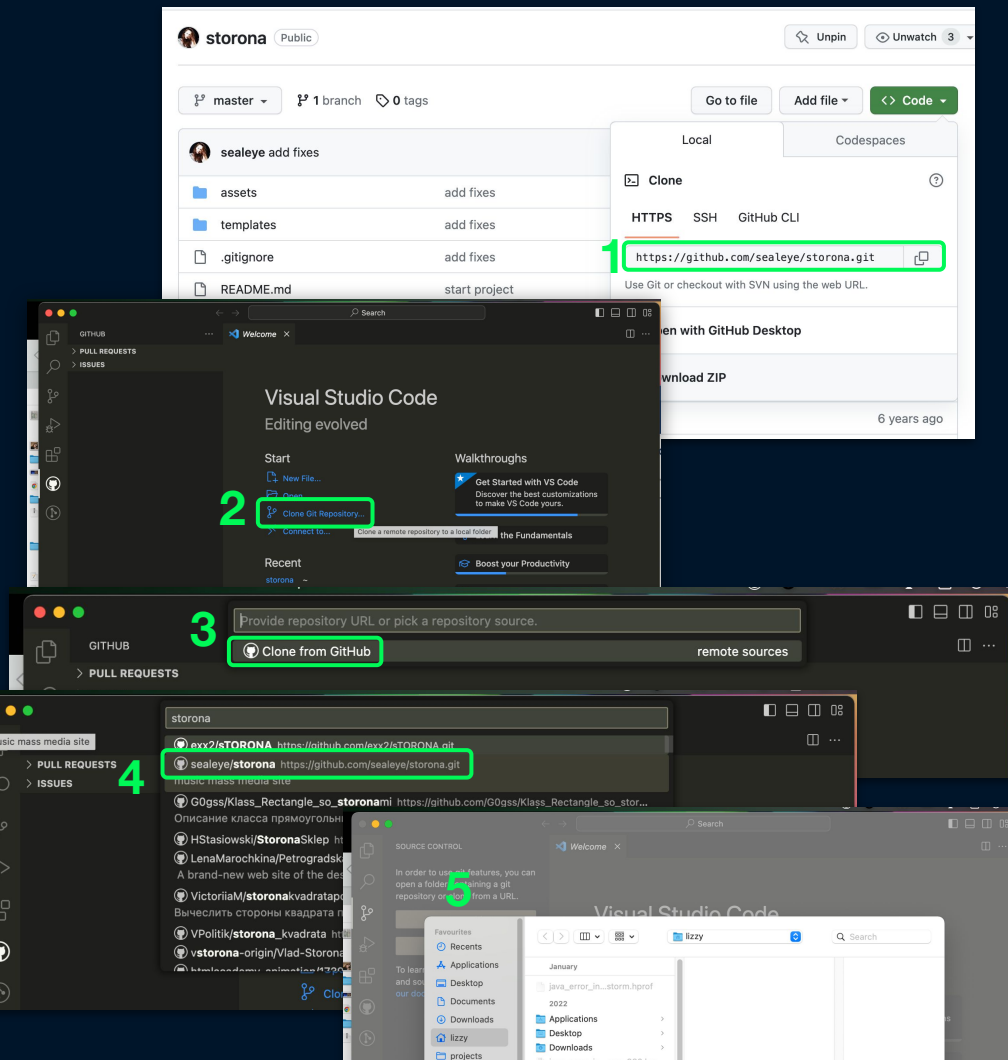
1. **Копируем** ссылку на репозиторий
2. В VS Code в правой панели выбираем пункт **Source Control** и там нажимаем на кнопку **Clone Repository**
3. **Вставляем** ссылку в верхнюю строку
4. **Выбираем папку**, где на компьютере будет лежать папка с кодом



## 2.3 Clone

Используем **VS Code** и плагин **GitHub**

1. **Копируем** ссылку на репозиторий
2. На главной странице редактора кода выбираем пункт **Clone Git Repository**
3. Жмем на пункт **Clone from GitHub**
4. **Выбираем проект**, который нужно клонировать
5. **Выбираем папку**, где на компьютере будет лежать папка с кодом

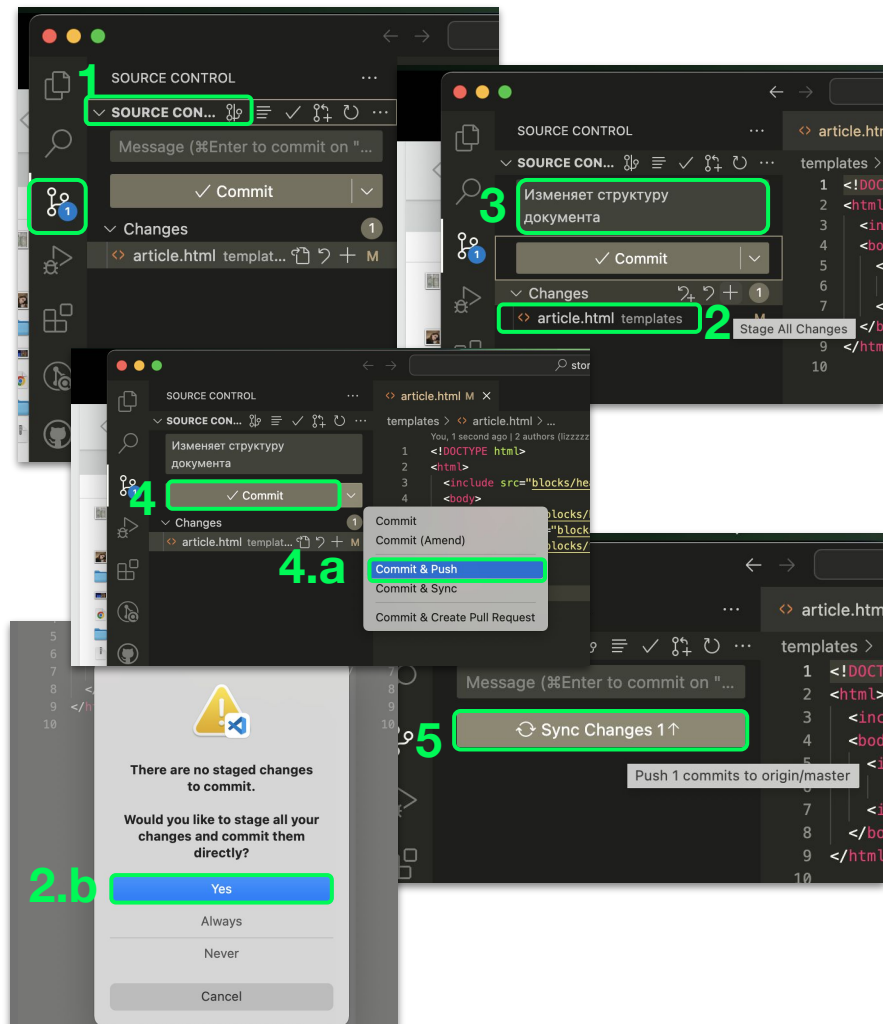


3

# Изменение кода

1. В правой панели выбираем пункт **Source Control** и открываем первый пункт – **Source Control** и видим файлы, которые изменились
2. Выбираем изменения, которые должны попасть в КОММИТ
  - a. На данном этапе можно просто нажимать на кнопку **+ пункта Changes** и выберутся все изменения
  - b. Если вы забудете выбрать изменения, то появится предупреждение и смело нажимайте **Yes**
3. Вводим сообщение в поле **Message** (оно должно отвечать на вопрос “Что делает этот коммит?”)
4. Нажимаем на кнопку **Commit**
  - a. Можно сразу выбрать пункт **Commit & Push**
5. Если не выбрали в предыдущем пункте, то сейчас ждем на кнопку **Sync Changes** и произойдет действие **Push**

Можно открыть свой репозиторий на **github.com** и убедиться, что все изменения дошли.

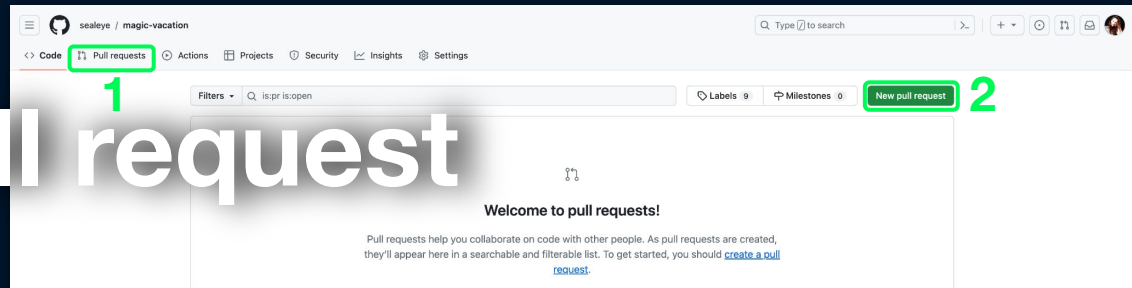


4

# Создание pull request

Используем **github.com**

1. Открываем репозиторий на github и выбираем вкладку **Pull requests**
2. Жмем на кнопку **New pull request**
3. Показывается панель с коммитами, которые войдут в pull request, тут нужно нажать на кнопку **Create pull request**
4. В описании пишем какие изменения были сделаны и нажимаем на кнопку **Create pull request**
5. Проверяем созданный Pull request



## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: `htmlacademy-animation/145...` base: `master` head repository: `sealeye/magic-vacation` compare: `master`

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

**Create pull request**

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: `htmlacademy-animation/145...` base: `master` head repository: `sealeye/magic-vacation` compare: `master`

✓ **Able to merge.** These branches can be automatically merged.

**Update README.md**

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits by maintainers

**Create pull request**

Remember, contributions to this repository should follow its [contributing guidelines](#).

## Update README.md #2

**Open** sealeye wants to merge 1 commit into `htmlacademy-animation:master` from `sealeye:master`

Conversation 0 Commits 1 Checks 0 Files changed 1



sealeye commented now

No description provided.

**Update README.md**

sealeye marked this pull request as ready for review now

Add more commits by pushing to the `master` branch on `sealeye/magic-vacation`.

**This branch has not been deployed**  
No deployments

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Процесс работы на курсе



1. Каждый работает с репозиториями в своем аккаунте
2. Один проект = один репозиторий

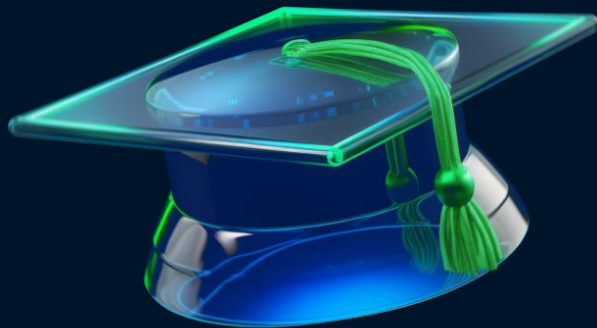
## Когда задание сделано:

1. Пишем код
2. Делаем commit, указываем что в нем сделано
3. Делаем push – отправка изменений в репозиторий
4. Изменение статуса в jira на in review. В комментарии к задаче указывайте ссылку на последний коммит.
5. Если задача оказывается в статусе reopen, процесс повторяется с пункта 1.



# Что мы сегодня выучили

1. Что такое система контроля версий и зачем она нужна
2. Что такое `git` и `github` 
3. Как работать с репозиториями
4. Основные термины `git` 
5. Процесс работы с `git` над проектом на курсе



**B** Academy  
**RO**



**THANK YOU!**

**QUESTIONS?**