

Git

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Изучить работу по веткам, научиться разрешать конфликты (resolve conflicts) через VSCode и в браузере. Научиться делать revert и reset.

ПЛАН ЗАНЯТИЙ

Повторение базовых команд git и основных определений

Revert

Создание веток (локальных и удаленных)

Reset

Разбор основных сценариев командной работы, Pull request

Локальный vs удаленный репозитории

- локальный (расположен непосредственно в памяти компьютера разработчика, в нем происходит разработка и фиксация изменений, после чего можно отправить на удаленный репозиторий)
- удаленный (находится на сервере, может быть приватным – доступным ограниченному числу лиц, и публичным – open source)

Про папки и репозитории

Если папка — это то, к чему мы все привыкли как пользователи компьютеров, то репозиторий — это что-то новое, что нужно создать, инициализировать. Сам по себе репозиторий без наших указаний не появляется. Репозиторий в наших задачах — это папка, над которой были произведены некоторые действия, и Git в ней начинает выполнять свои задачи, например:

- **отслеживать изменения файлов;**
- **хранить информацию о ветках.**

Важно! Репозиторий не возникает сам по себе, его нужно создать

Как понять, в репозитории мы находимся или в папке?

Самый простой способ это сделать — набрать в терминале команду «git status». Если в ответ вы увидите ошибку «fatal: not a git repository (or any of the parent directories): .git», значит, в терминале вы вызываете команду не из репозитория, а из обычной папки. Если вы увидели что-то другое, то вы находитесь в репозитории или внутри одной из папок, которая находится в нем.

Важно! Репозиторий отслеживает изменения во всех вложенных в него папках.

Если вы сделаете репозиторием корневую папку на диске C (не делайте этого!), то весь ваш диск станет репозиторием и Git будет пытаться отслеживать все изменения на этом диске. Создаем репозитории очень аккуратно.

Как можно создать репозиторий?

Чаще всего на начальных этапах рассматривают два способа создания репозитория:

- Если мы находимся в папке (!) и хотим сделать из нее репозиторий, то вызываем команду «**git init**», и эта папка становится репозиторием.
- Если мы хотим клонировать репозиторий из GitHub на свой ПК, то мы пользуемся командой «**git clone**».

(При этом обратите внимание: не нужно пользоваться командой «git init», команда clone не только скачивает файлы из интернета, но и инициализирует репозиторий в скачанной папке. На самом деле она делает сильно больше, но нам важно, что в скачанной папке у нас уже будет репозиторий и никак дополнительно инициализировать его не надо).

Внимательно следим за тем, из какой папки вы вызываете команды

Терминал всегда показывает, в какой папке вы сейчас находитесь, но первое время студенты чаще смотрят на то, какая папка открыта в визуальном интерфейсе редактора (например, VSCode), а не на то, что написано в терминале. Обращайте, пожалуйста, внимание на название папки, которая указана в приглашении к вводу команд терминала. До тех пор, пока вы не привыкнете к работе с терминалом, внимательно следите за тем, что вы создаете репозитории только во вновь созданных для урока папках. Не нужно создавать репозитории из рабочего стола или других больших папок.

Внимательно следим за тем, из какой папки вы вызываете команды

В данном примере терминал открыт в папке `git_example_04`

```
○ aliserkhamidov@MacBook-Pro-2 git_example_04 % █
```

Папка vs локальный репозиторий

Когда вы создаете локальный репозиторий, у вас в папке появляется новая **скрытая папка с названием «.git»**. Это специальная папка, в которой хранится все, что необходимо для работы системы контроля версий. **Если вы удалите эту папку, то потеряете всю историю, которую Git успел сохранить, но при этом превратите ваш репозиторий обратно в папку.**

Итак, чтобы из репозитория снова сделать папку, достаточно всего лишь удалить скрытую папку «.git». При этом вы потеряете историю, которую собрал Git (все коммиты, ветки и т. п.), но файлы в самой папке останутся в том же виде, в котором они были в момент удаления папки «.git».

(Если вы работаете на Windows, включите отображение скрытых файлов и папок, так как папка .git скрытая. Это можно сделать в верхнем меню. На маке достаточно нажать **'cmd + shift + .'**

Ветвление

Почти каждая система контроля версий в той или иной форме поддерживает ветвление.

Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от нее, не вмешиваясь в основную линию.

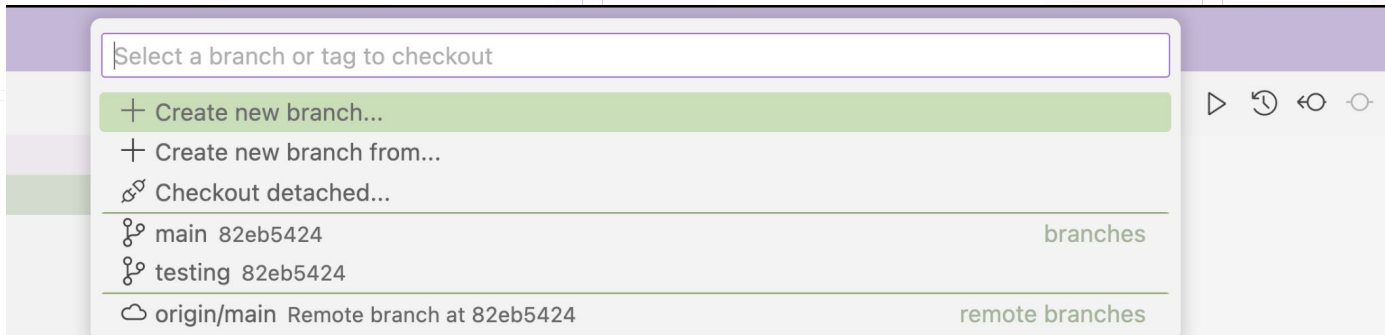
```
// команда для создания ветки с названием testing  
git branch testing
```

Ветвление

Посмотреть доступные ветки можно введя команду **git branch**

```
aliserkhamidov@MacBook-Pro-2 git_example_04 % git branch
* main
  testing
```

Также это можно сделать через интерфейс VSCode, если нажать в левом нижнем углу на название ветки:



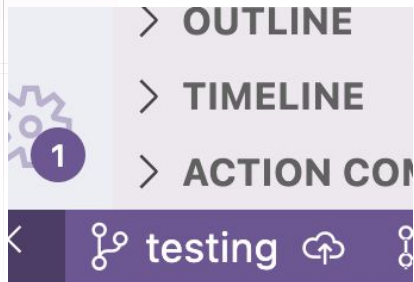
Переключение между ветками

Переключиться на ветку можно при помощи команды:

git checkout имя_ветки

- aliserkhamidov@MacBook-Pro-2 git_example_04 % git checkout testing
Switched to branch 'testing'
- aliserkhamidov@MacBook-Pro-2 git_example_04 % █

Текущая ветка указана в левом нижнем углу:



Ветвление: локальная ветка

Мы можем сразу создать ветку и переключиться на нее

Для этого нам нужно добавить флаг `-b`

В данном примере мы создали ветку с названием **mybranch** из текущей ветки (HEAD) и переключились на нее

```
git checkout -b mybranch
```

Удаленная ветка

Флаг **-u** требуется добавлять

только при первом пуше:

git push -u origin имя_ветки

Эта команда установит

удаленную ветку,

соответствующую нашей

локальной

// переключаемся на ветку

git checkout **-b** mybranch

// сделаем изменения кода, которые мы желаем сделать

// после этого:

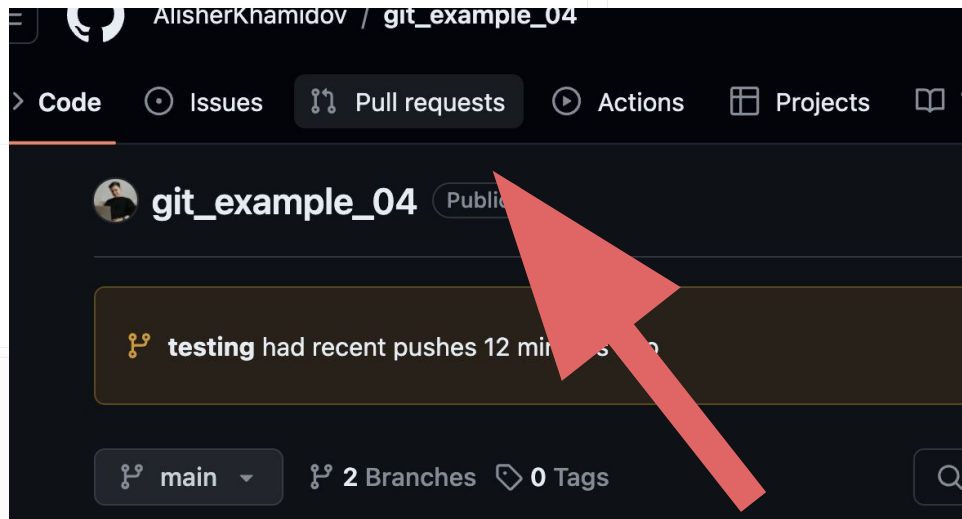
git add .

git commit -m 'add input for email'

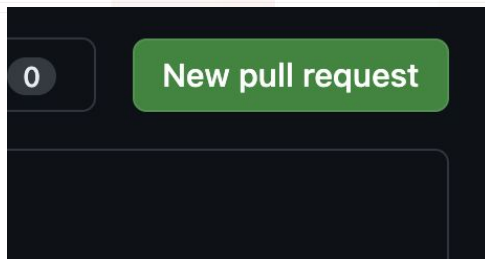
git push -u origin mybranch

Pull request

Заходите в репозиторий на
github во вкладу **Pull requests**

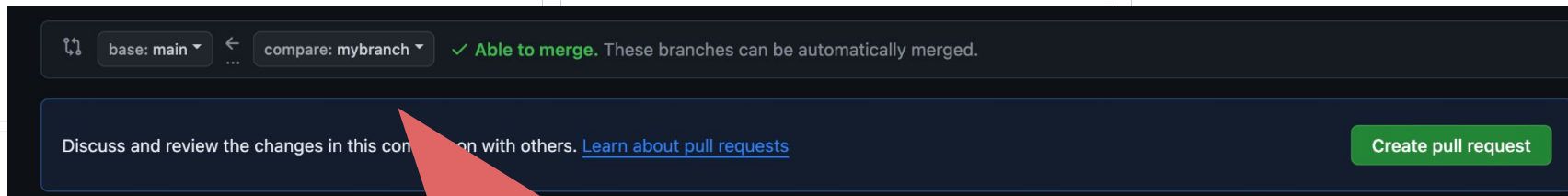


Нажимаете на кнопку New
pull request



Pull request

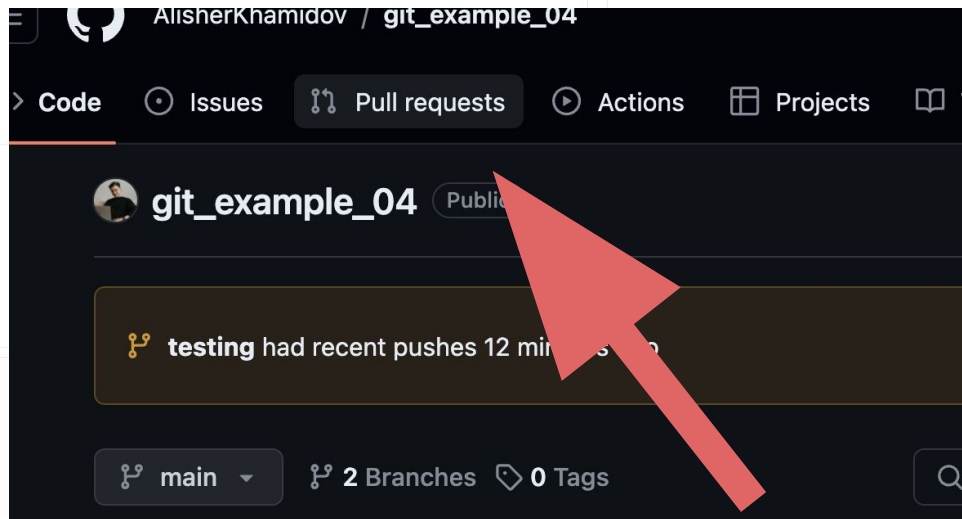
Выберите из какой ветки в какую вы собираетесь перенести изменения



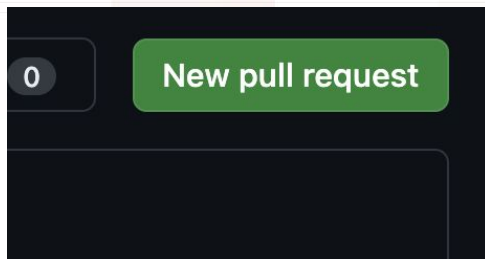
В данном примере мы переносим изменения из ветки **mybranch** в ветку **main**

Pull request

Заходите в репозиторий на
github во вкладу **Pull requests**



Нажимаете на кнопку New
pull request



После этого можно осуществить merge

Обратите внимание на то, кто именно в вашей команде будет одобрять слияние.



Require approval from specific reviewers before merging

[Rulesets](#) ensure specific people approve pull requests before they're merged.

Add rule



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](#).

Resolve conflict

При слиянии веток вы можете столкнуться с конфликтами.

Конфликт возникает, когда программа не может выбрать между двумя версиями кода и “нуждается” в нашей подсказке в том, что мы хотим оставить.

Resolve conflict

Конфликт можно разрешить нажав кнопку Resolve conflict

Add more commits by pushing to the mybranch branch on AlisherKhamidov/git_example_04.



This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

index.html

Resolve conflicts

Merge pull request



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

Resolve conflict

После этого вам будет предложен код, в котором специальными символами будут показаны разные версии кода, можно оставить нужный участок кода или оба. При этом строки со специальными символами необходимо удалить.

```
10      <p>Change 3</p>
11      <p>Change 8</p>
12      <<<<<< mybranch
13      <p>My branch</p>
14      <p>My branch 2</p>
15      =====
16      <p>Change from someone</p>
17      >>>>>> main
18  </body>
```

```
11      <p>Change 8</p>
12      <p>My branch</p>
13      <p>My branch 2</p>
14      <p>Change from someone</p>
15  </body>
16  </html>
```

Commit merge

Mark as resolved

Resolve conflict VSCode

Настройте VSCode, чтобы разрешать конфликты.

Откройте настройки, в поисковой строке введите слово conflict.

Поставьте галочку в настройке: Open the merge editor for files that are currently under conflict

Git: Merge Editor



Open the merge editor for files that are currently under conflict.



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH