

Практика работы с Git и GitHub



Установите Git и SourceTree для своей ОС

<https://www.sourcetreeapp.com/>

При работе с Windows, Git можно по ссылке

<https://git-scm.com/download>

Если работаем на Windows, то используйте программу Git Bash, которая запускает терминал

Обязательная настройка

Используйте свое имя, фамилию и адрес эл. почты.

```
$ git config --global user.name "Artem Starostenko"
```

```
$ git config --global user.email artemstarostenko65@gmail.com
```

Без этих данных Git не позволит нам делать комиты

Начинаем отслеживать историю

- Создайте пустую директорию `practice-git` и перейдите в нее
- Инициализируйте репозиторий, используя команду `git init`. Убедитесь, что создалась директория `.git`

Добавим файл

- `$ echo "My first line in project" > file.txt`
- `$ git status` # понимаете ли вы почему файл помечен, как `untracked`?
- `$ git add file.txt` # Добавляем файл в Index
- `$ git status`
- `$ git diff --cached` # смотрим проиндексированные изменения

Сохраним изменение в истории

- `$ git commit -m "My first commit"`
- `$ git log #` проверьте, что комит есть в истории
- `$ git show #` посмотрите информацию о последнем комите

Добавим еще один КОМИТ

- `$ echo "London is the capital of GB" >> file.txt`
- `$ git diff # непроиндексированные изменения, сравните с git diff --cached`
- `$ git commit -am "Capital added" # удалось ли нам сделать комит без добавления в Index или в Index мы все же добавляем изменения перед комитом?`
- `$ git log # проверяем в истории, сделали ли мы КОМИТ`

Исправьте предыдущий комит

В последнем комите мы зафиксировали добавление строки “London is the capital of GB” в файл file.txt, но мы забыли добавить в файл, что “Moscow is the capital of Russia”. Исправьте последний комит (не добавляйте новый) и в сообщении комитета укажите, что была добавлена не одна столица, а несколько.

Ответы даны на следующем слайде.

ОТВЕТЫ

- `$ echo "Moscow is the capital of Russia" >> file.txt`
- `$ git add file.txt`
- `$ git commit --amend # поменяйте сообщение,`
что добавлена была не одна столица, а
несколько

СМОТРИМ ИСТОРИЮ

`$ git log` # убедитесь, что комита по-прежнему 2 и нам действительно удалось изменить последний КОМИТ, не создавая НОВЫЙ

```
commit 66bca8c1208dae1a067c805fb4ad7d970eaabd89
Author: Artem Starostenko <artemstarostenko65@gmail.com>
Date: Tue Nov 22 12:23:32 2016 +0300
```

Capitals added

```
commit bf79e90c5a4d5e82171ba6bc81c847e958015846
Author: Artem Starostenko <artemstarostenko65@gmail.com>
Date: Tue Nov 22 12:17:50 2016 +0300
```

My first commit

Проверяем изменения в последнем комите

```
$ git show
```

```
commit 66bca8c1208dae1a067c805fb4ad7d970eaabd89
```

```
Author: Artem Starostenko <artemstarostenko65@gmail.com>
```

```
Date: Tue Nov 22 12:23:32 2016 +0300
```

Capitals added

```
diff --git a/file.txt b/file.txt
```

```
index ee2ae0c..ddc9cc9 100644
```

```
--- a/file.txt
```

```
+++ b/file.txt
```

```
@@ -1,3 @@
```

```
My first line in project
```

```
+London is a capital of GB
```

```
+Moscow is a capital of Russia
```

Добавим Францию

Добавьте в конец файла `file.txt` строку “Paris is the capital of France” и зафиксируйте эти изменения, сделав комит. В сообщении комита укажите “Another capital was added”.

Используйте команды `git log` и `git show`, чтобы проверить сделанный комит.

Отменим второй комит

- Найдите hash 2-ого комита из git log
- `$ git revert 1f829c0` # используйте первые символы своего хеша (рекомендуется указывать первые 7 символов), чтобы отменить комит. Также можно воспользоваться HEAD~1
- Должен возникнуть конфликт

Решаем конфликт

`$ git status` #смотрим, где у нас произошел конфликт.

`$ cat file.txt` # смотрим конфликтующие строки

My first line in project

<<<<<< HEAD

London is the capital of GB

Moscow is the capital of Russia

Paris is the capital of France

=====

>>>>>> parent of 1f829c0... Capital added

Git помечает спец символами, в каких строчках у нас конфликт

Какие изменения относятся к отменяемому комиту?

`$ git show HEAD~1` # смотрим, какие строки нам нужно удалить, чтобы отменить при решении конфликта, чтобы решить нашу первичную задачу – отменить изменения второго комита.

`@@ -1 +1,3 @@`

My first line in project
+London is the capital of GB
+Moscow is the capital of Russia

Решение конфликта

Шаг 1: отменить
изменения комита

```
My first line in project
<<<<<< HEAD
London is the capital of GB
Moscow is the capital of Russia
Paris is the capital of France
=====
>>>>>> parent of 1f829c0... Capital added
```

Шаг 2: привести
содержимое файла
в нужное состояние

```
My first line in project
<<<<<< HEAD
Paris is the capital of France
=====
>>>>>> parent of 1f829c0... Capital added
```


Продолжаем отменять КОМИТ

- `$ git add file.txt` # говорим, что мы разрешили конфликт
- `$ git revert --continue`
- `$ git log` # посмотрите, что создан новый КОМИТ
- `$ git show` # посмотрите, что поменялось в последнем комите

Работаем с .gitignore

- `$ echo "this is a garbage file" | tee garbage1.tmp garbage2.tmp`
создайте два файла с одинаковым расширением.
- `$ git status #` проверьте, что Git видит эти файлы
- Создайте файл `.gitignore` и укажите в нем, чтобы гит игнорировал все файлы с расширением `tmp`, т.к. мы не хотим их отслеживать
- `$ git status #` файлы с расширением `.tmp` должны исчезнуть
- Закоммитьте `.gitignore`

Задание на понимание Index

Добейтесь следующего результата. В результате выполнения команды `git status` видим, что `file.txt` присутствует в двух секциях одновременно: `changes to be committed` и `changes not staged for commit`.

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
modified:   file.txt
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
modified:   file.txt
```

Результат закомитьте

- Чтобы рабочая директория была чистая

Работа с ветками

- Создание и удаление веток
- Мердж веток
- Перемещение (rebase) веток
- Клонирование удаленного репозитория
- Pull и Push в удаленный репозиторий

Создадим ветку

- `$ git branch first` # Создайте ветку с именем first
- `$ git branch` # Посмотрите список локальных ВЕТОК

```
$ git branch # звездочкой будет помечена текущая ветка
first
* master
```

- `$ git checkout first` # переключиться на ветку first

Создадим еще одну ветку

- `$ git checkout -b second` # создаем и переключаем на ветку одной командой (сравните, как было с веткой first)
- `$ git status` # также подскажет, на какой ветке мы находимся
- `$ git branch` # список локальных веток

Изменим ветку second

```
$ echo "Let's change the branch" > branch.txt
```

Закомитьте изменения с сообщением **"Branch was changed"**

Сделаем коммит на первой ветке

Переключитесь на ветку first, создайте файл first.txt с любым содержимым и закомитьте изменения.

Постарайтесь сделать самостоятельно. Ответы даны на следующем слайде.

```
$ git checkout first
Switched to branch 'first'

$ echo "And this branch too" > first.txt

$ git add first.txt

$ git commit -m "First branch changed"
[first 2e54f37] First branch changed
1 file changed, 1 insertion(+)
create mode 100644 first.txt
```

Смотрим граф изменений

```
$ git log --all --decorate --oneline --graph
```

Создайте alias для данной команды, она нам еще пригодится.

```
$ git config --global alias.l 'log --all --decorate --oneline --graph'
```

```
$ git l
```

```
* 2e54f37 (HEAD -> first) First branch changed
| * f279210 (second) Branch changed
| /
* 17d8431 (master) .gitignore added
* f9e503c Revert "Capitals added"
* de140fc France added
* 66bca8c Capitals added
* bf79e90 My first commit
```

Смержим ветки

```
$ git checkout master # перейдем в мастер
```

```
Switched to branch 'master'
```

```
$ ls
```

```
$ git merge first
```

```
Updating 17d8431..2e54f37
```

```
Fast-forward
```

```
first.txt | 1 +
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 first.txt
```

```
$ ls # появился файл first.txt
```

Заметьте, что при слиянии не создалось дополнительного комита, переместился только указатель ветки master. Как называется данный тип слияния? (ответ есть в лекции)

```
* 2e54f37 (HEAD -> master, first) First branch changed
| * f279210 (second) Branch changed
| /
* 17d8431 .gitignore added
* f9e503c Revert "Capitals added"
* de140fc France added
* 66bca8c Capitals added
* bf79e90 My first commit
```

Отребейзим ветку second

```
$ git checkout second
```

```
Switched to branch 'second'
```

```
$ git rebase master
```

```
First, rewinding head to replay your work on top of it...  
Applying: Branch changed
```

Ветка second поменяла свое основание

```
$ git l
* b77f3f6 (HEAD -> second) Branch changed
* 2e54f37 (master, first) First branch changed
* 17d8431 .gitignore added
* f9e503c Revert "Capitals added"
* de140fc France added
* 66bca8c Capitals added
* bf79e90 My first commit
```

- Линейная история
- Ветки master и first совпадают
- Ветка second на 1 комит опережает мастер

Удаляем ветку

```
$ git branch -d first
```

```
Deleted branch first (was 2e54f37).
```

```
$ git branch
```

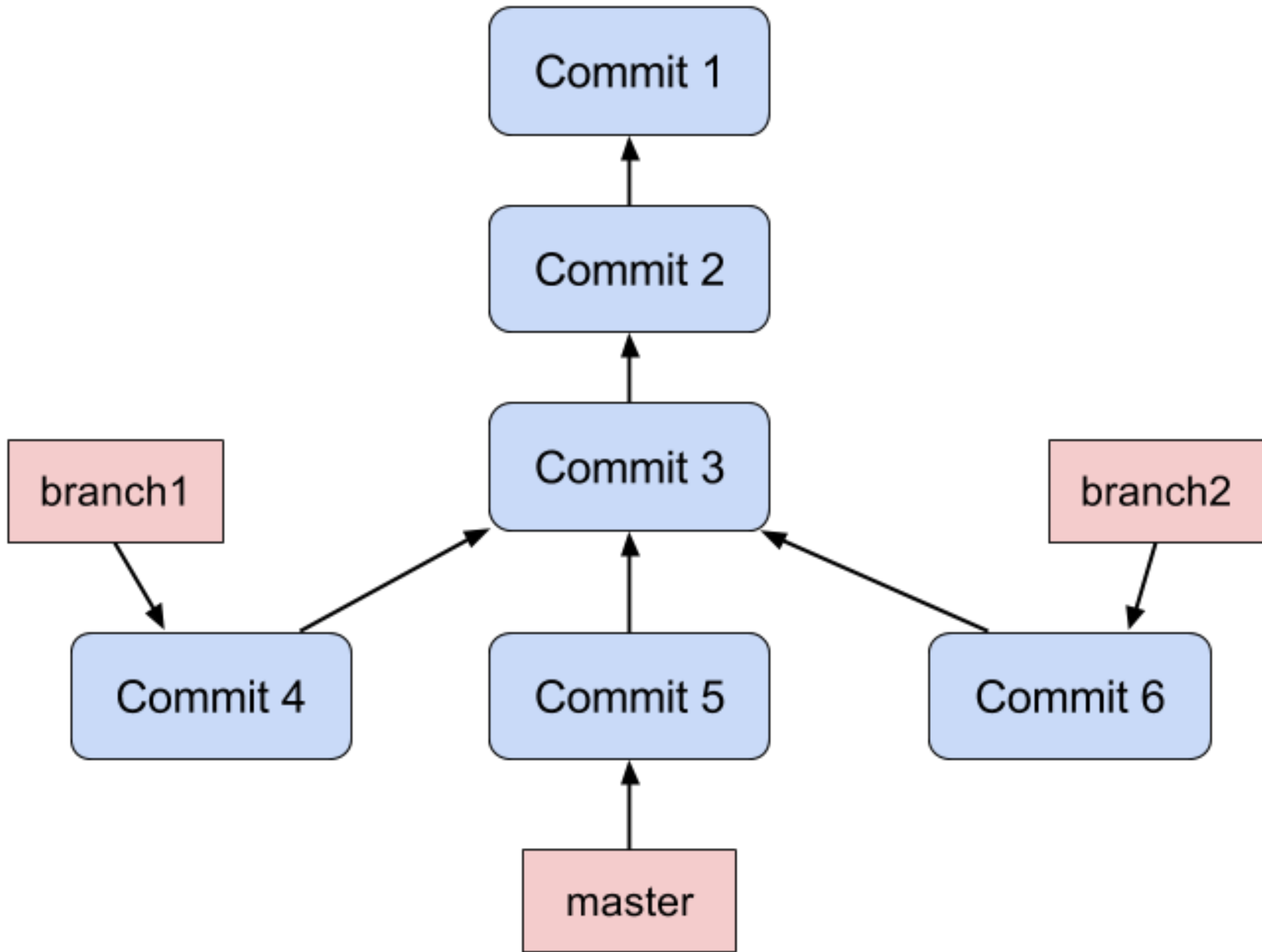
```
master
```

```
* second
```


Задание

- Создайте новую директорию task-branches, в ней создайте git репо и “нарисуйте” следующую картинку.

Примечание к следующему слайду: надписи в синих прямоугольниках (например, “Commit 2”) означают сообщение комита, а не последовательность выполнения самих комитов. Master, branch1, branch2 - названия веток.



Регистрация на GitHub

Рекомендуется подключить двухфакторную аутентификацию, чтобы обезопасить ваш аккаунт (для этого можно использовать мобильное приложение Authy, Google Authenticator)

Добавить двухфакторную аутентификацию можно, перейдя в Settings -> Security

Two-factor authentication

Status: **Off** ✕

Set up two-factor authentication

Two-factor authentication provides another layer of security to your account.

Регистрация на GitHub


Добавьте SSH ключ для своего аккаунта, чтобы можно было работать с удаленным репозиторием через SSH.

Инструкции по созданию пары ключей для вашей ОС можно посмотреть [здесь](#).

Инструкции по добавлению публичного ключа для своего аккаунта можно посмотреть [здесь](#).


Скопируйте свои репозитории на GitHub


Создайте пустой GitHub репозиторий practice-git-1 для нашего первого локального репозитория (practice-git)

Owner:  Artemmkin ▾ / Repository name: practice-git-1 ✓

Great repository names are short and memorable. Need inspiration

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ Initialize this repository with a README

Скопируйте свои репозитории на GitHub

Перейдем в папку practice-git, где выполняли первые задания, и настроим информацию об удаленном репозитории

```
$ git remote add origin git@github.com:<UserName>/practice-git-1.git
```

Пушим все ветки локального репозитория в удаленный

```
$ git push origin -u --all
```

-u опция позволяет задать для локальной ветки дефолтную удаленную ветку для синхронизации, так что команды git pull/push будут работать на данной локальной ветке без доп аргументов.

Скопируйте свои репозитории на GitHub

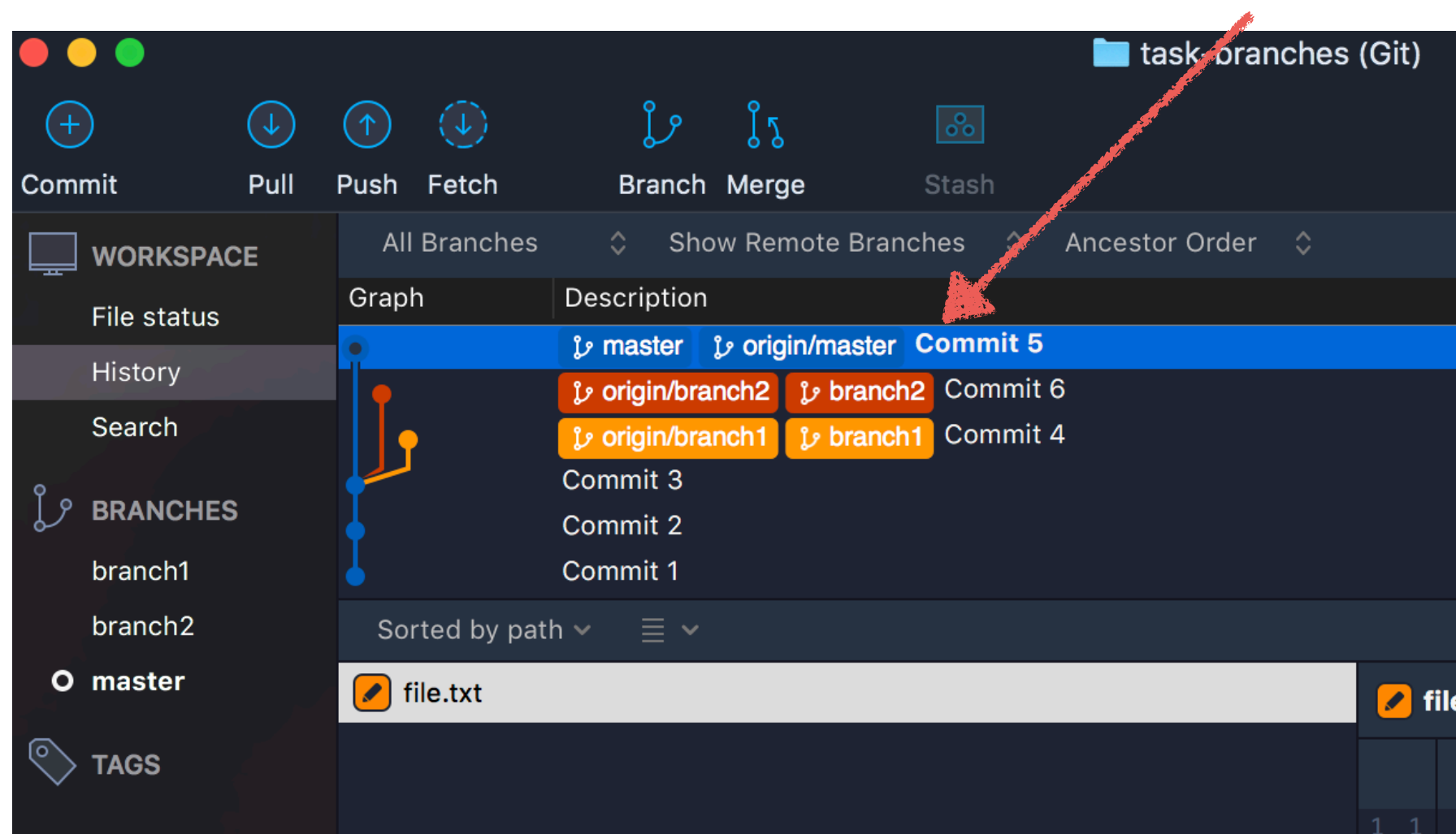
Создайте GitHub репозиторий practice-git-2 для задания с ветками (task-branches), запушьте туда изменения по аналогии с предыдущим репозиторием.

Добавьте пользователя Artemmkin в collaborators. Для этого зайдите в настройки репозитория

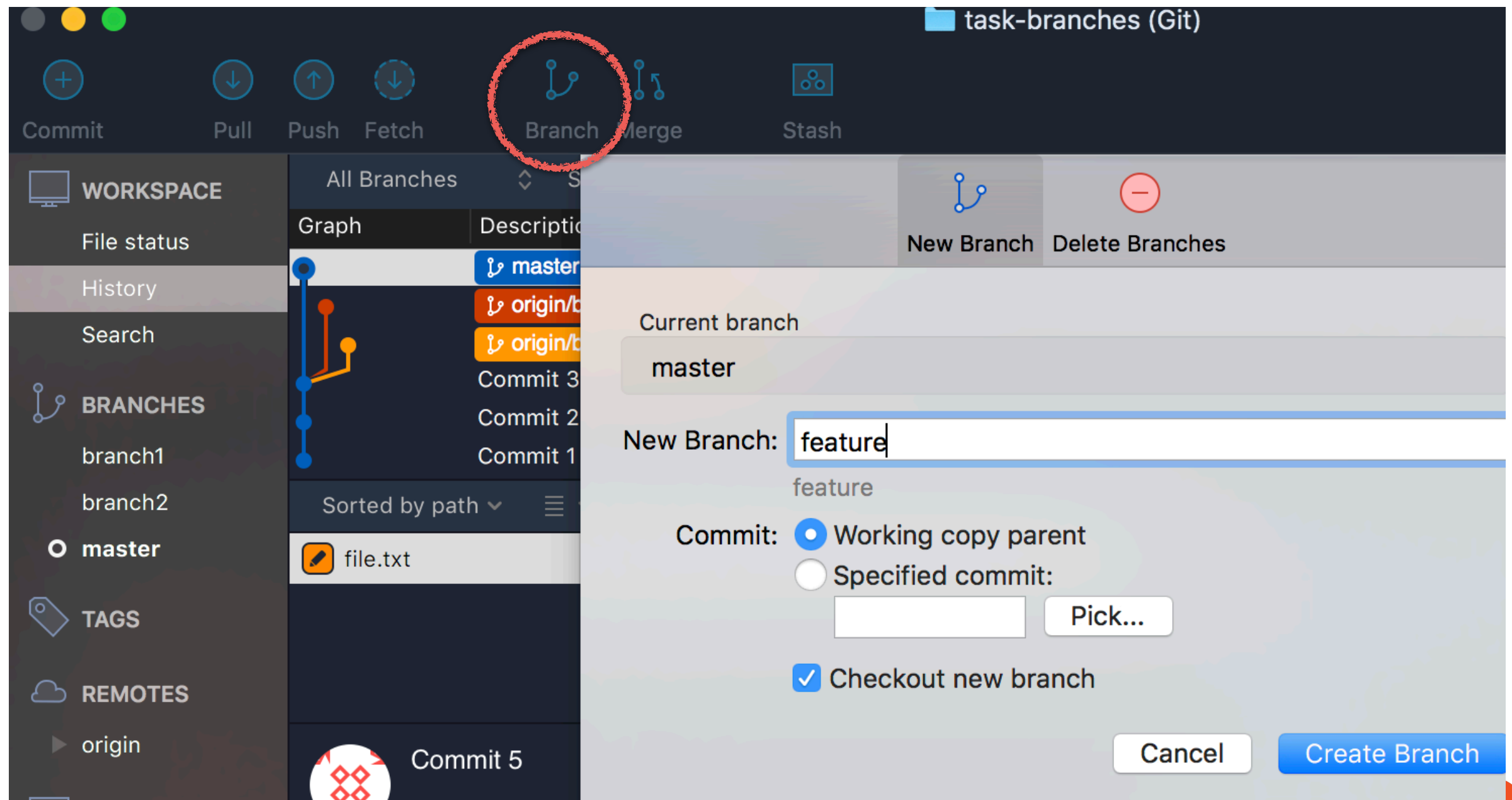
The screenshot displays the GitHub repository settings interface. At the top, there's a navigation bar with tabs: Code, Issues (0), Pull requests (0), Projects (0), Wiki, Settings, and Insights. Below this, on the left, is a sidebar with links: Options, Collaborators (selected), Webhooks, Integrations & services, and Deploy keys. The main content area is titled 'Collaborators' and includes a sub-header 'Push access to the repository'. A message states: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' Below this is a search section titled 'Search by username, full name or email address' with a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' A search input field contains the text 'Artemmkin', and to its right is a button labeled 'Add collaborator'. A red hand-drawn box highlights the search input and the 'Add collaborator' button.

SourceTree

Добавляем проект task-branches в SourceTree. Два раза кликаем на ветку master, чтобы в нее перейти.



Создаем ветку feature

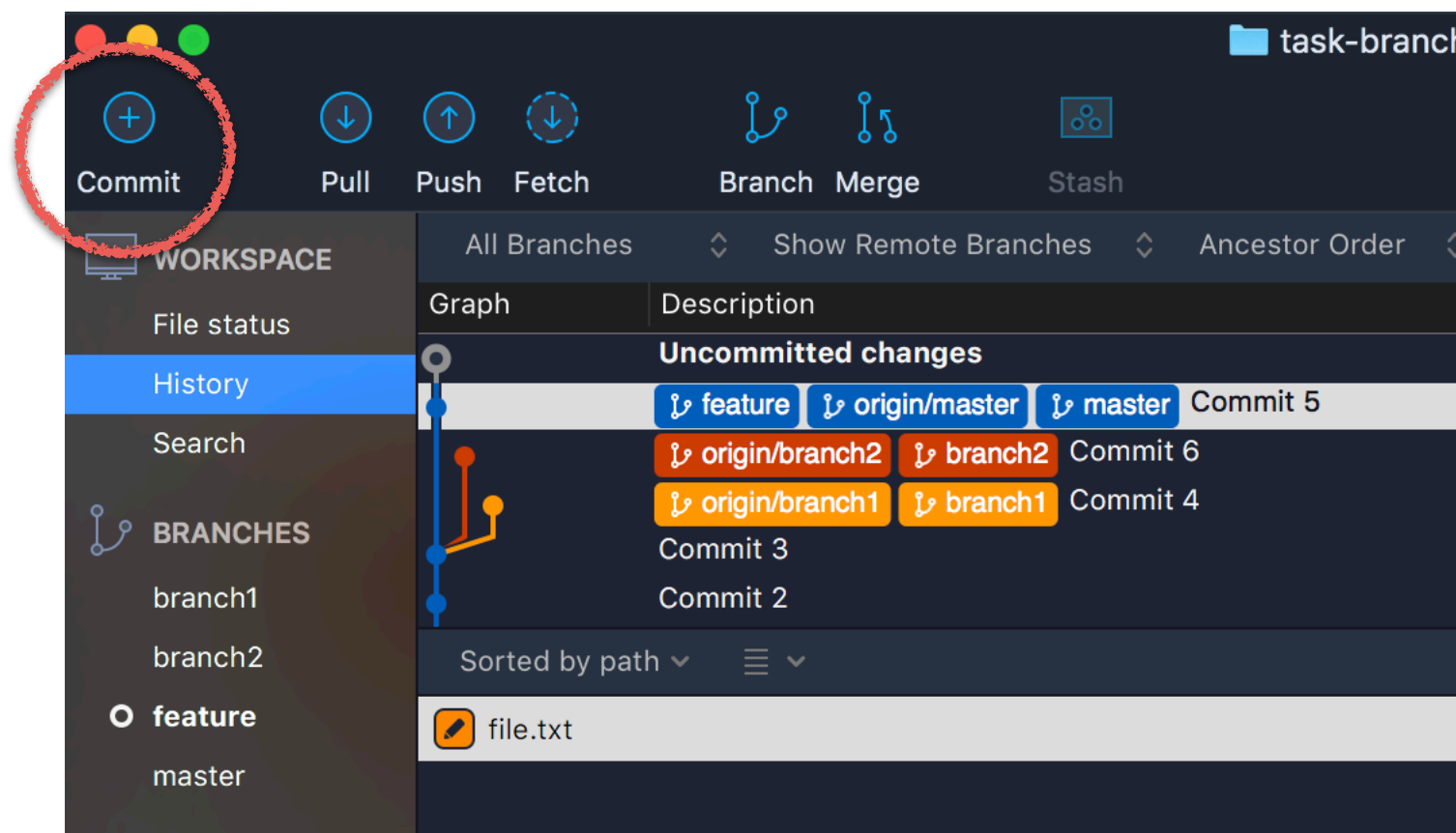


Проверяем в терминале, что ветвь создалась

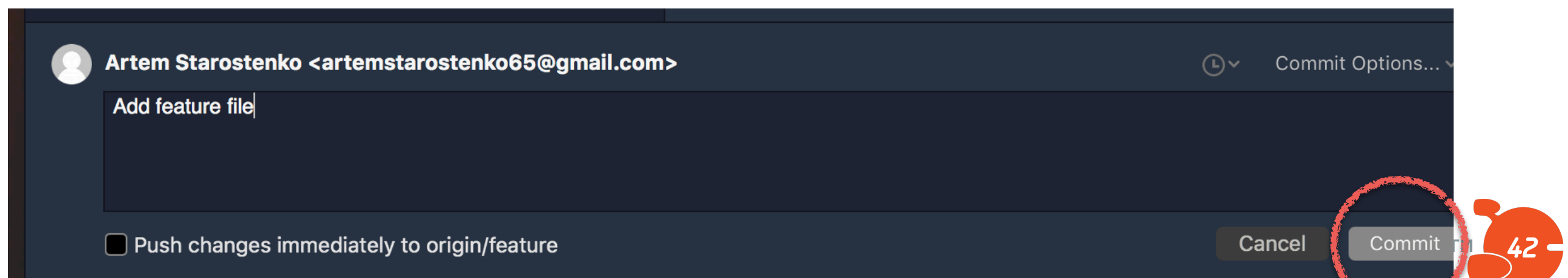
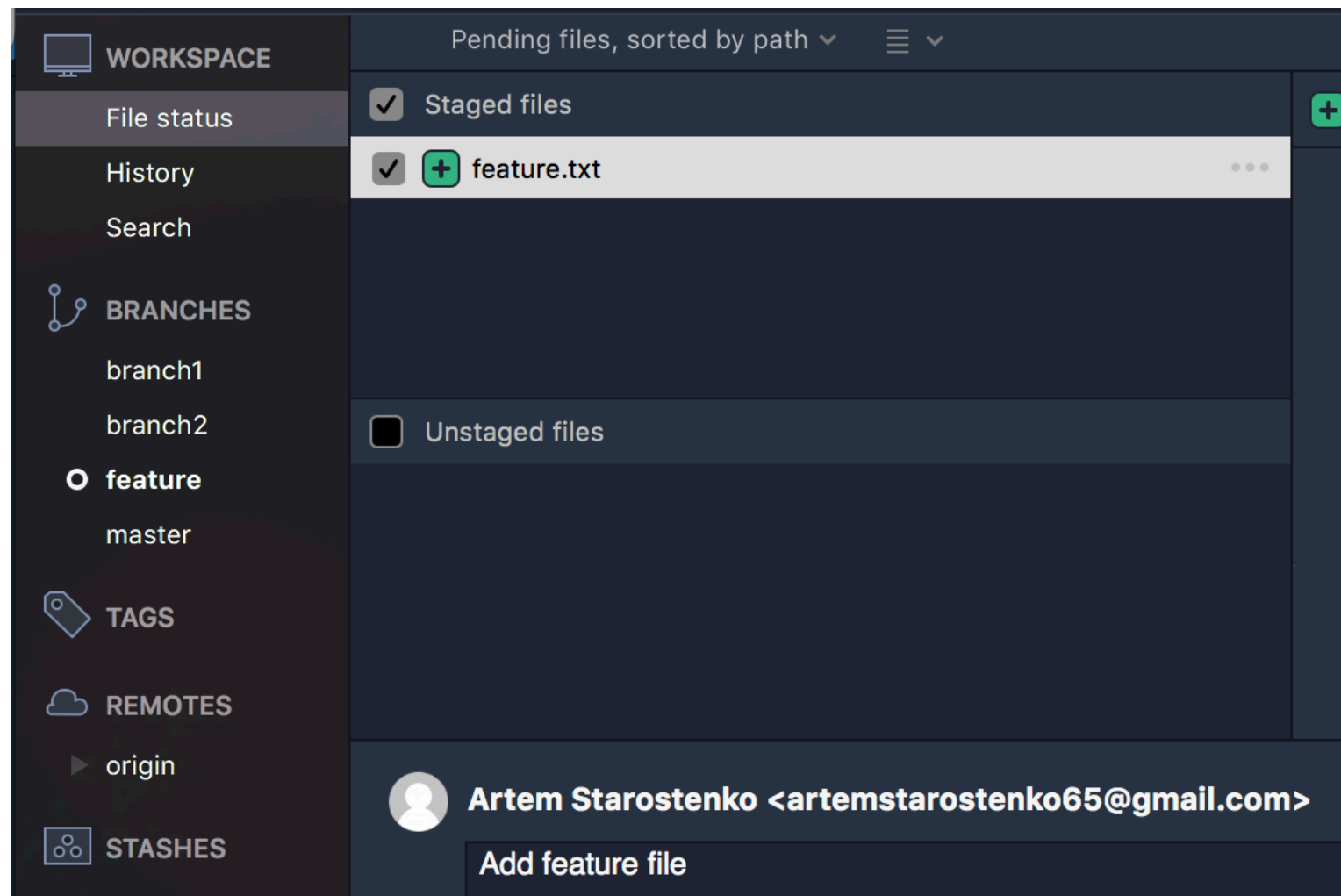
```
$ git branch  
branch1  
branch2  
* feature  
master
```

Делаем коммит из SourceTree

Создайте пустой файл feature.txt. В SourceTree отобразится информация о незакоммиченных изменениях. Нажмите Commit, чтобы сделать коммит.



Ставим галку напротив файла feature в секции Unstaged - файл будет переведен в Stage. И нажмим Commit под сообщением.



Добавление изменений по частям

Перейдите на ветку feature, кликнув по ней два раза в SourceTree. В терминале вы должны видеть, с какой веткой вы работаете. Для этого пользователям линукс и макбуков, можно воспользоваться темами oh my zsh

```
artemkin@macpro:~/task-branches <feature>$
```

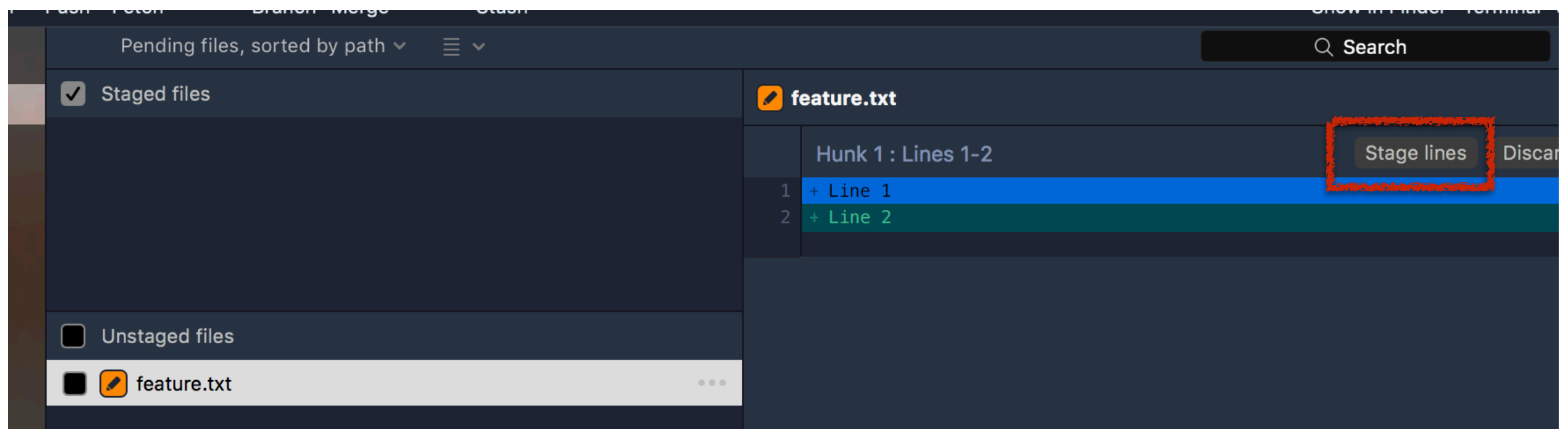
Добавьте в файл feature.txt две любые строки.
Например:

```
$ echo "Line 1" >> feature.txt
```

```
$ echo "Line 2" >> feature.txt
```

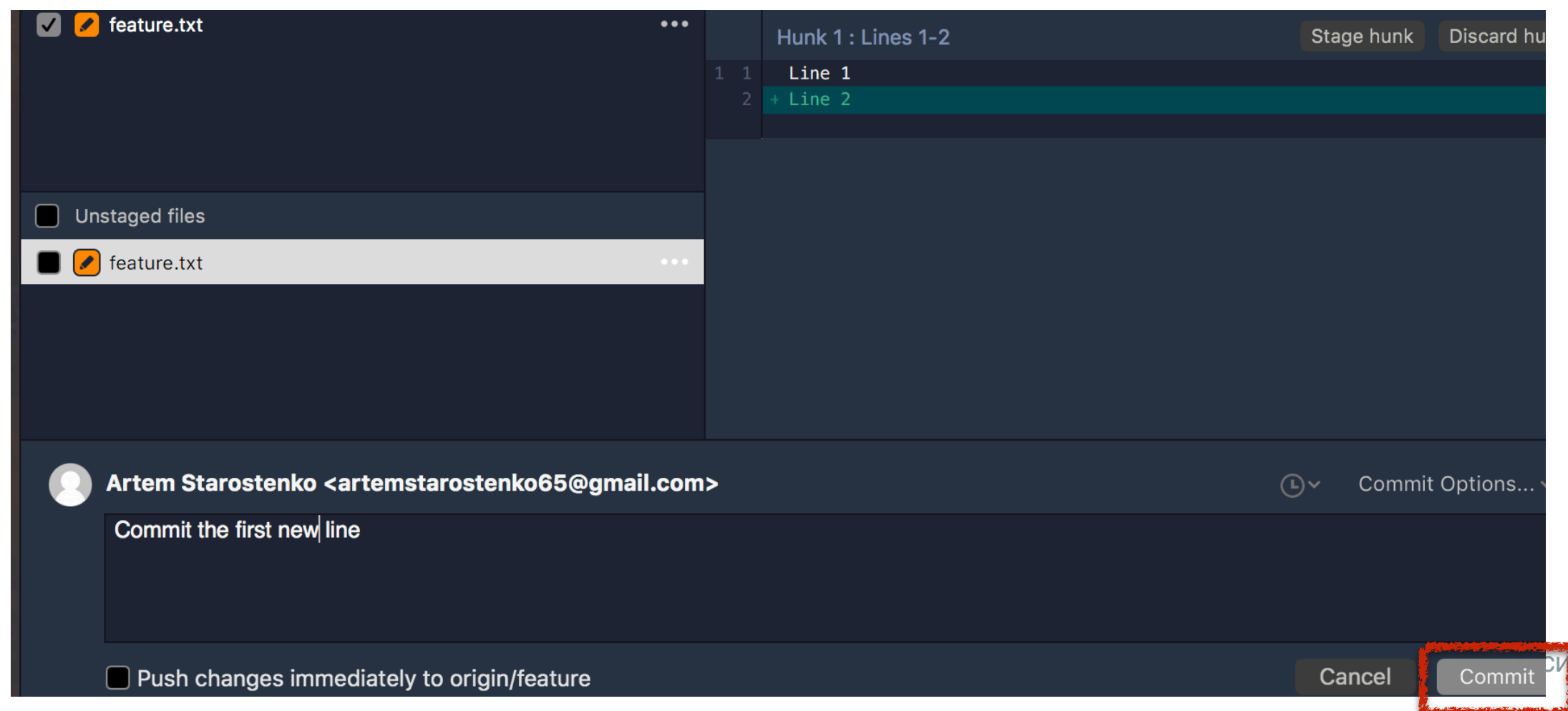
Добавляем изменения по частям

Нажмите Commit в SourceTree. Добавьте в Stage первую добавленную строку. Для этого кликните на нее и нажмите Stage lines



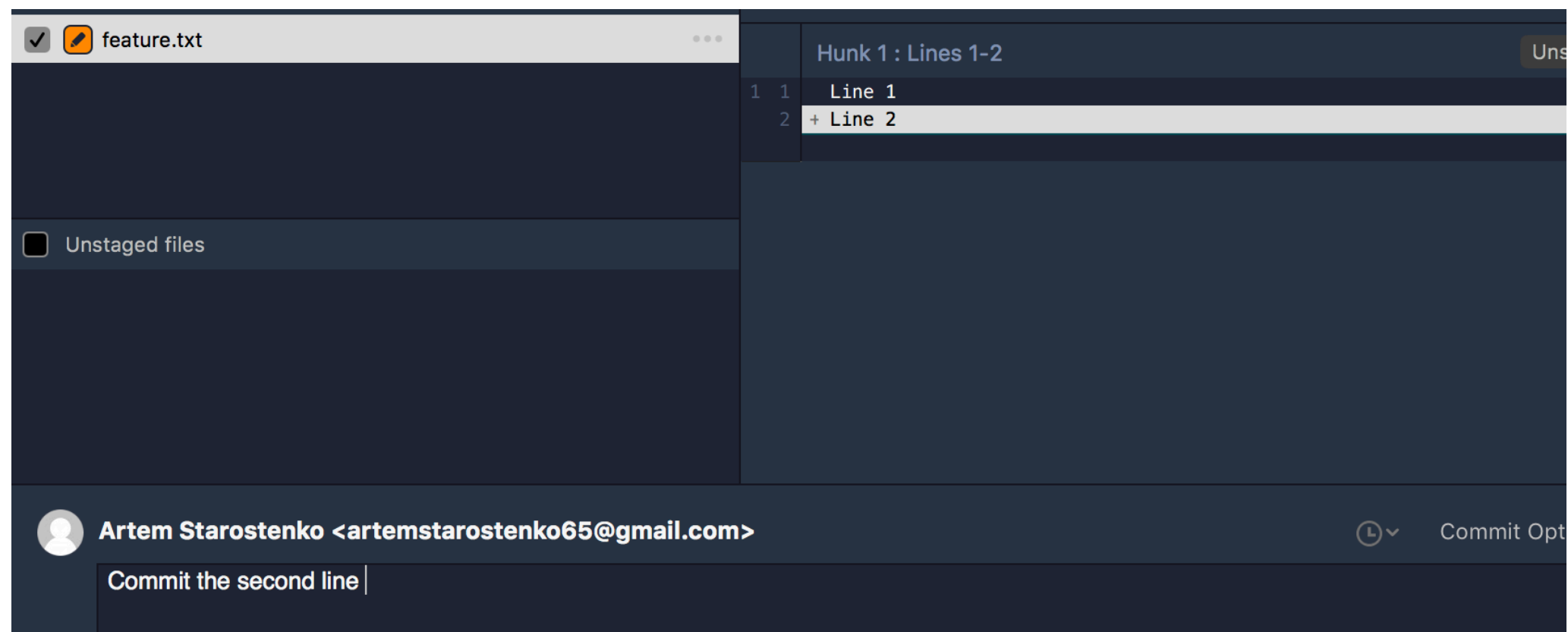
Добавляем изменения по частям

Напишите сообщение, что была закомичена первая строка. И сделайте комит.



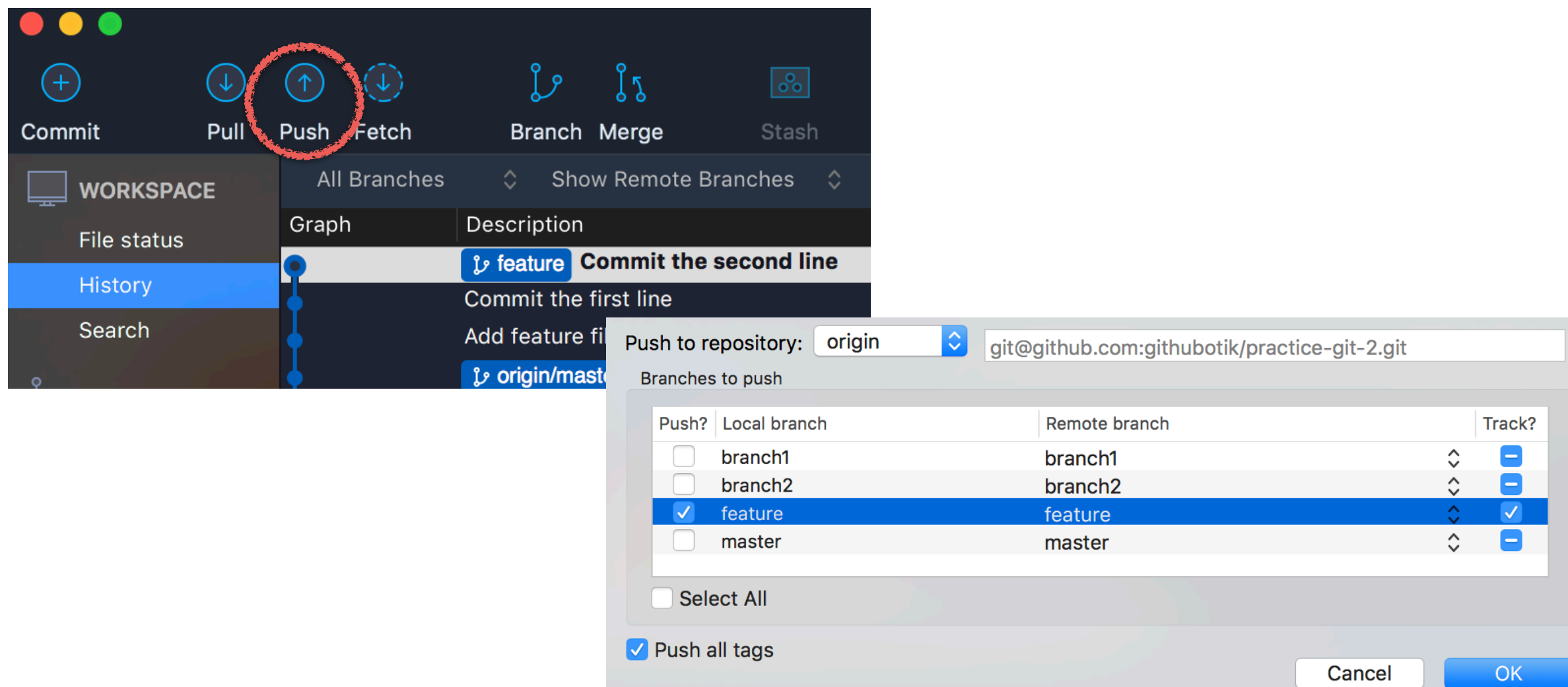
Добавляем изменения по частям

Теперь сделайте по аналогии комит для второй добавленной строки.



Создание PR

Сделайте push ветви feature в удаленный репозиторий на GitHub.



Создание PR

На страницу GitHub вашего репозитория practice-git-2. Создайте новый запрос на мерж ветки feature в master. Не забудь добавить reviewers (см. след. слайд)

4 commits 4 branches

Branch: master ▾ **New pull request**

Artemmkin Commit 5

file.txt

Open a pull request

Create a new pull request by comparing changes across two branches. If you need 1

base: master ▾ ... compare: feature ▾ **✓ Able to merge.** These branches

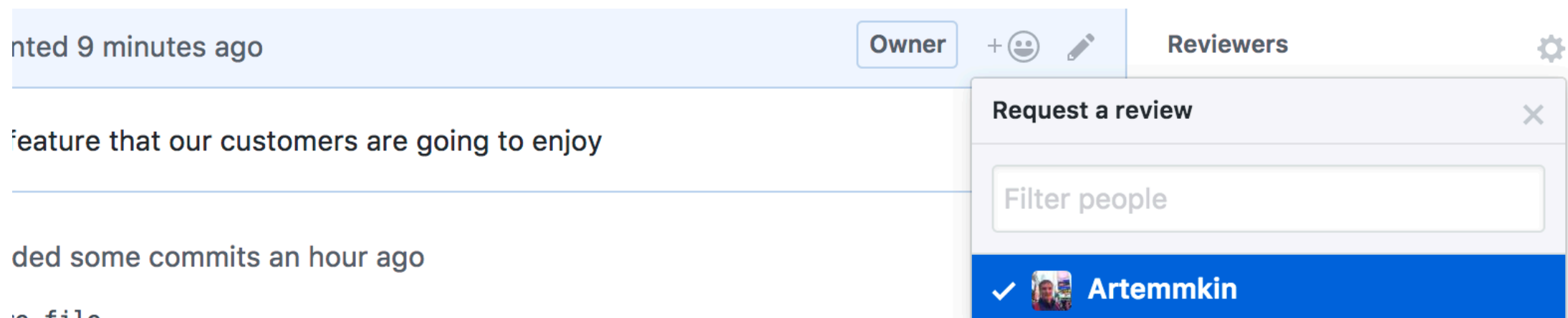
Add new feature for our website

Write Preview **AA** ▾ **B** *i* “ <> 🔗

This is a new cool feature that our customers are going to enjoy |

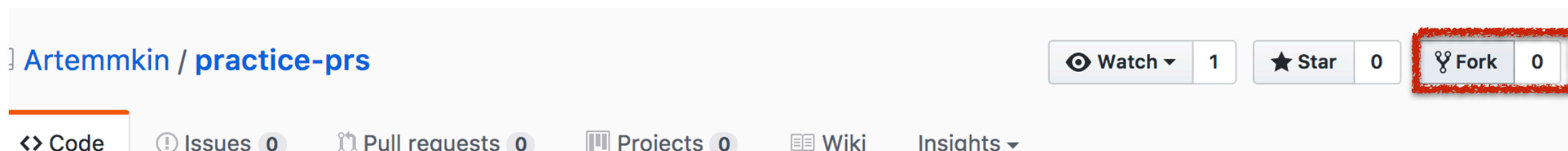
Создание PR

Добавьте в Reviewers пользователя Artemmkin, чтобы попросить его посмотреть ваши наработки прежде чем делать мерж веток. Попросите еще кого-нибудь из нашей учебной группы сделать ревью. Вы, в свою очередь, можете сделать ревью (добавить комментарии) их наработок. После получения одобрений от всех reviewers сделайте мерж.



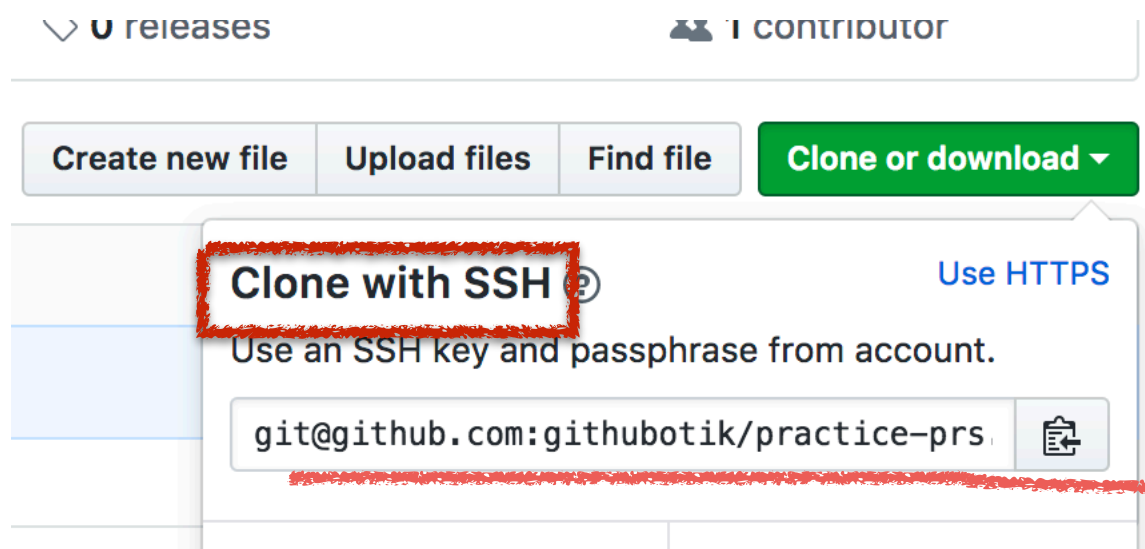
Делаем PR к чужому репозиторию

Сделайте Fork тестового репозитория



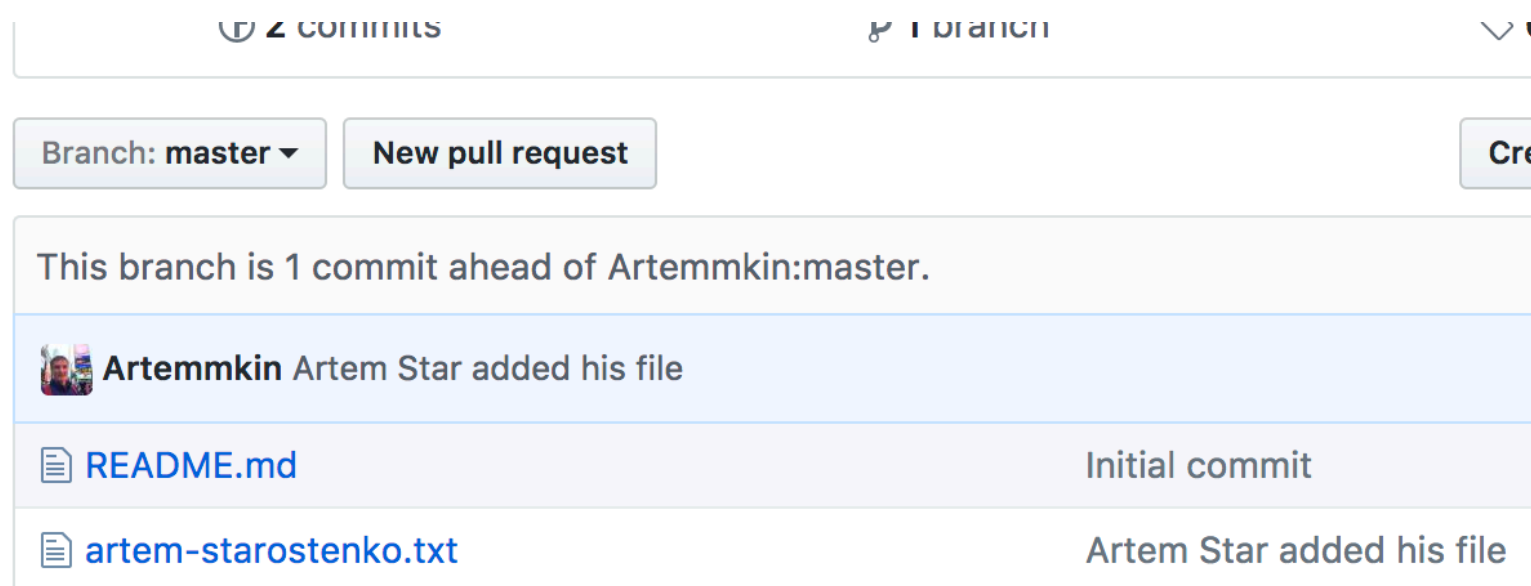
Склонируйте репозитория на локальную машину

`$ git clone <url>`



Делаем PR к чужому репозиторию

При клонировании репозитория будет создана папка с названием проекта. Создайте в этой папке пустой файл. Имя файла должно быть в формате имя-фамилия.txt. Например: artem-starostenko.txt. Закомитьте этот файл. И сделайте пуш в свой форкнутый репозиторий на GitHub (команда `$ git push`).



Делаем PR к чужому репозиторию

Теперь нам нужно перейти на страницу репозитория, форк которого мы создавали, и создать новый пул реквест.

The screenshot shows a GitHub repository interface. At the top, there are tabs for '2 commits', '1 branch', and a dropdown menu. Below these, there is a 'Branch: master' dropdown and a 'New pull request' button, which is highlighted with a red dashed box. To the right of this button is a 'Create pull request' button. Below the buttons, a message states 'This branch is 1 commit ahead of Artemmkin:master.' A commit by 'Artemmkin Artem Star' is listed, showing 'README.md' as the initial commit. Below the commit list, there is a 'Compare changes' section. It includes a description: 'Compare changes across branches, commits, tags, and more below. If you need to, you can also [compare across forks](#).' The 'compare across forks' link is highlighted with a red dashed box. Below this, there are two dropdown menus: 'base: master' and 'compare: master'. At the bottom, there is a green 'Create pull request' button and a text prompt: 'Choose different branches or forks above to discuss and review changes.'

Делаем PR к чужому репозиторию

Base fork - репозиторий, куда хотим влить изменения, head fork - ваш форкнутый репозиторий.

The image shows two screenshots of the GitHub web interface. The top screenshot shows the 'Create pull request' button and a dropdown menu for 'Choose a Head Repository'. The dropdown menu lists 'Artemmkin/practice-prs' (checked) and 'githubotik/practice-prs' (highlighted in blue). The bottom screenshot shows the 'base fork: Artemmkin/practice-prs' and 'head fork: githubotik/practice-prs' selected, with a green checkmark indicating 'Able to merge'. Below this, a comment box is visible with the text 'Artem Star added his file' and a 'Write' button.

base fork: Artemmkin/practice-prs base: master ... head fork: Artemmkin/practice-prs compare: master

Create pull request Choose different branches or forks

Choose a Head Repository

Filter repos

✓ Artemmkin/practice-prs

githubotik/practice-prs

base fork: Artemmkin/practice-prs base: master ... head fork: githubotik/practice-prs compare: master

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

Artem Star added his file

Write Preview

AA B i “ <> 🔗 ⋮ 1/2 1/3 1/4 ↩ @ ★

Leave a comment

Несколько доп заданий для тех, кому интересно :)

Нужно ответить на вопросы, представленные на следующем слайде. Ответы вписать в файл с именем и фамилией, который мы в создали в предыдущем задании, и добавить изменения к только что созданному PR.

Вопросы со звездочкой

Вопросы касаются репозитория `zabbixar`, который можно скачать по данной ссылке.

1. Суммарное число комитов, которые меняли файл `Gemfile`?
2. Напишите дату когда, `Gemfile` был добавлен в репозиторий.
3. Какой пользователь в последний раз менял строки функции *`def proxies`* в файле `lib/zabbixar.rb`?
4. Напишите хеш комита (необязательно полностью), который имеет сообщение “Add Ruby 2.2 support”.
Плюсом будет, если предоставите ссылку на страницу GitHub данного комита.