



УРОК 16. PR В VISUAL STUDIO CODE

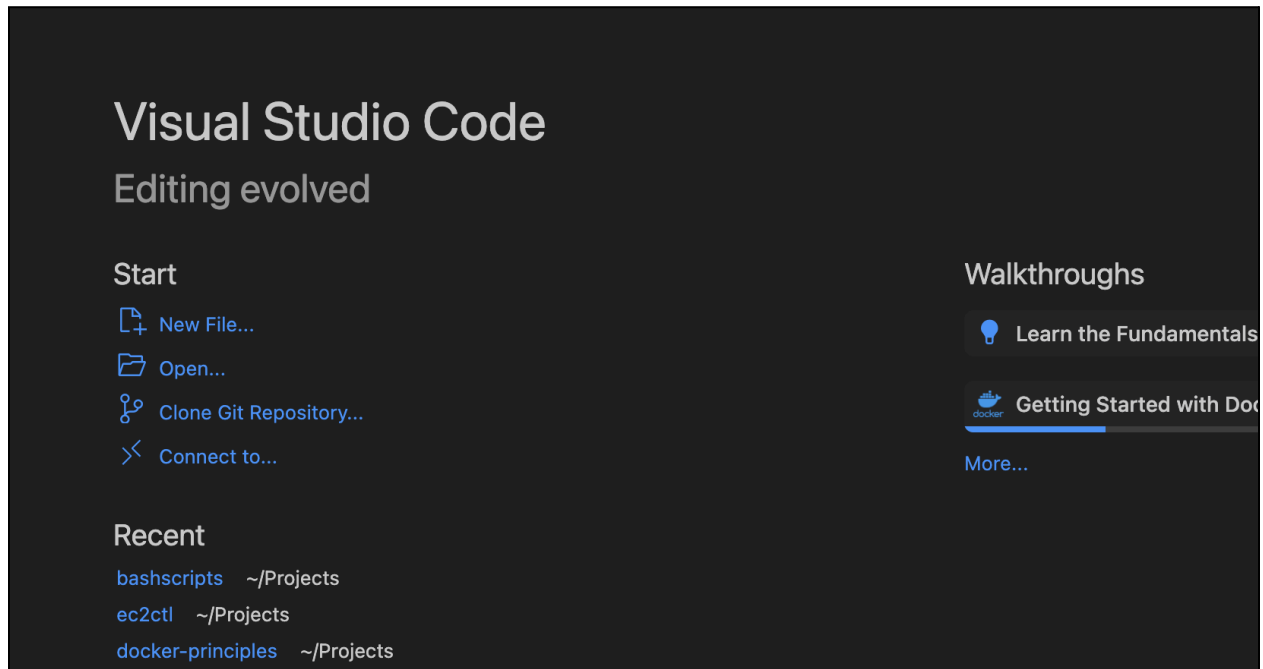
БАЗОВЫЙ ФУНКЦИОНАЛ VISUAL STUDIO CODE	2
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ	5
РАБОТА С РЕПОЗИТОРИЕМ	6
РАБОТА С PULL REQUEST	14
ОБНОВЛЕНИЕ ФОРКНУТОГО РЕПОЗИТОРИЯ	16



БАЗОВЫЙ ФУНКЦИОНАЛ VISUAL STUDIO CODE

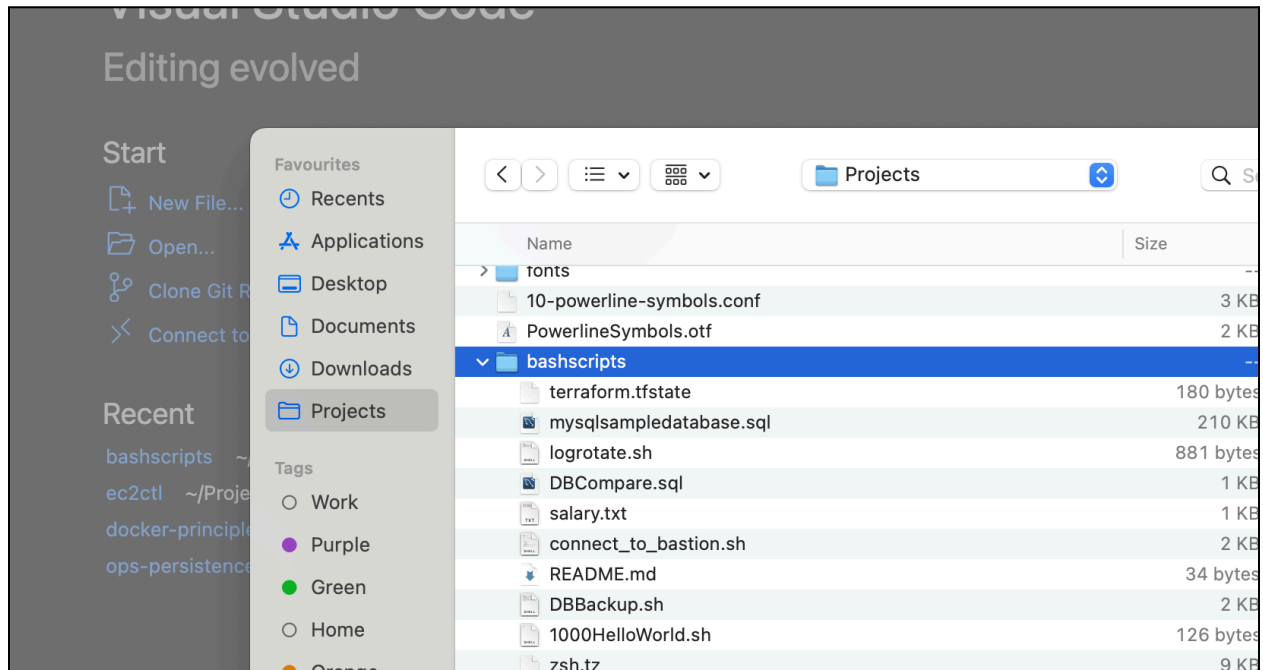
Рассмотрим базовый функционал Visual studio code и плагина Git.

Открываем редактор кода:



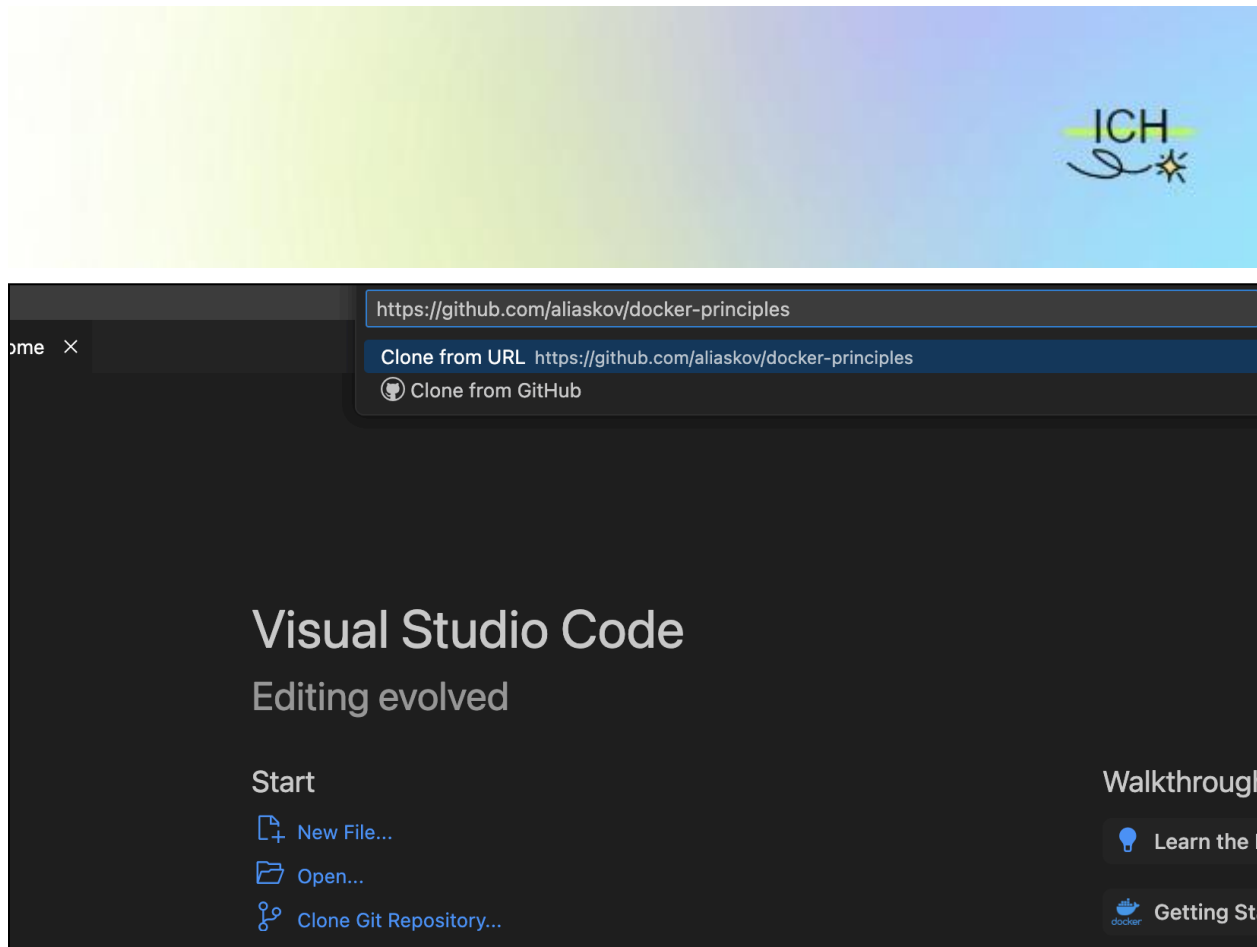
Мы можем открыть директорию, с существующим репозиторием, если вы клонировали или создали его ранее.

Для этого нажимаем на Open и выбираем директорию, с которой будем работать.

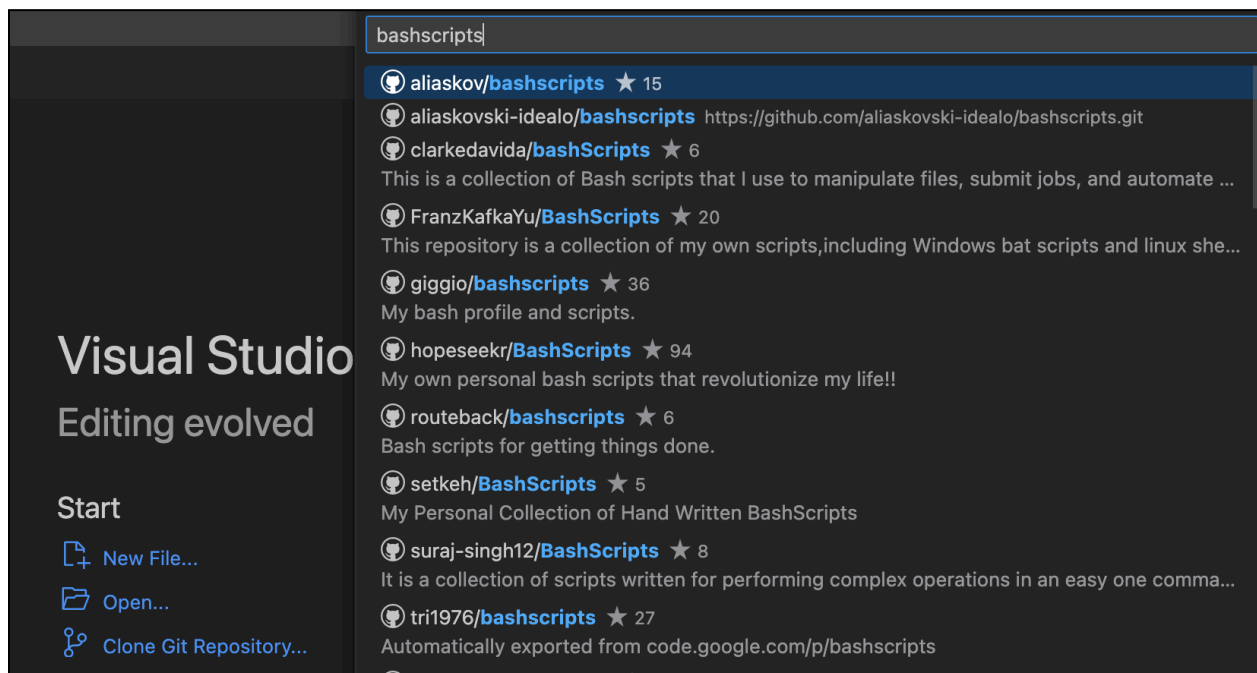


Как клонировать репозиторий при помощи Visual Studio Code

Альтернативно мы можем клонировать репозиторий при помощи Visual Studio Code и сразу открыть этот репозиторий - для этого нажимаем на Clone Git Repository и в строке поиска вводим ссылку на репозиторий .



Или ищем интересующий нас репозиторий:





ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Форкнуть репозиторий в github и клонировать его при помощи Visual Studio Code.



РАБОТА С РЕПОЗИТОРИЕМ

Разберем функционал плагина git в visual studio code. Добавим новый файл, закоммитим и запустим его.

Пример файла:

```
Unset
#!/bin/bash

# Версия операционной системы
os_version=$(cat /etc/os-release | grep "PRETTY_NAME" | cut -d '"' -f 2)

# Дата и время
current_date=$(date "+Y-%m-%d")
current_time=$(date "+H:%M:%S")

# Время работы системы
uptime_info=$(uptime -p)

# Загруженность системы
system_load=$(uptime | awk -F'[a-z]:' '{ print $2 }')

# Занятое дисковое пространство
disk_usage=$(df / | awk '{print $5}' | sed 's/%//')

# Топ процессы по использованию памяти
top_processes=$(ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -n 6)

# Количество процессов
process_count=$(ps -ef | wc -l)

# Количество пользователей
```



```
user_count=$(who | wc -l)

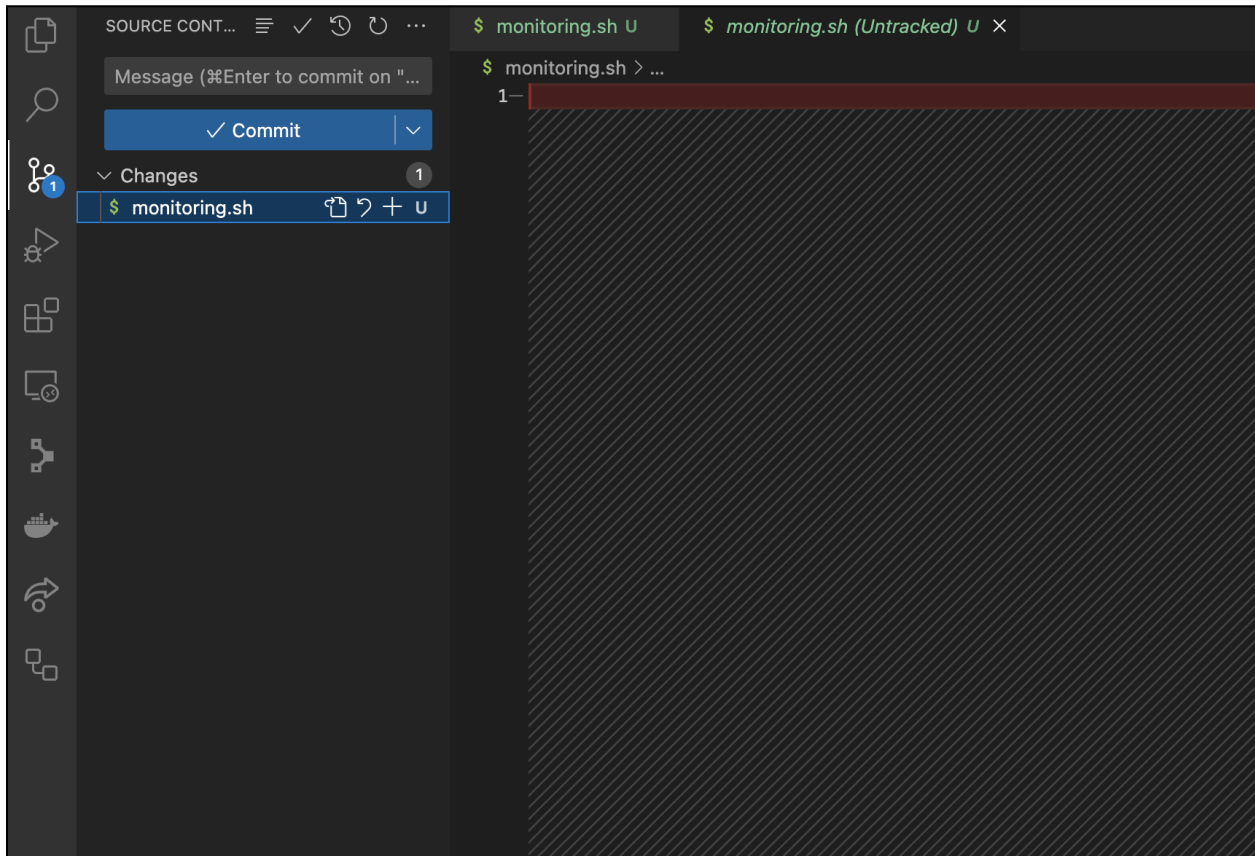
# Выводим отчет
echo "Отчет о системе"
echo "Версия операционной системы: $os_version"
echo "Дата: $current_date"
echo "Время: $current_time"
echo "Время работы системы: $uptime_info"
echo "Загруженность системы: $system_load"
echo "Занятое дисковое пространство: $disk_usage"
echo "Топ процессы по использованию памяти:"
echo "$top_processes"
echo "Количество процессов: $process_count"
echo "Количество пользователей: $user_count"
```

Добавим файл monitoring.sh:

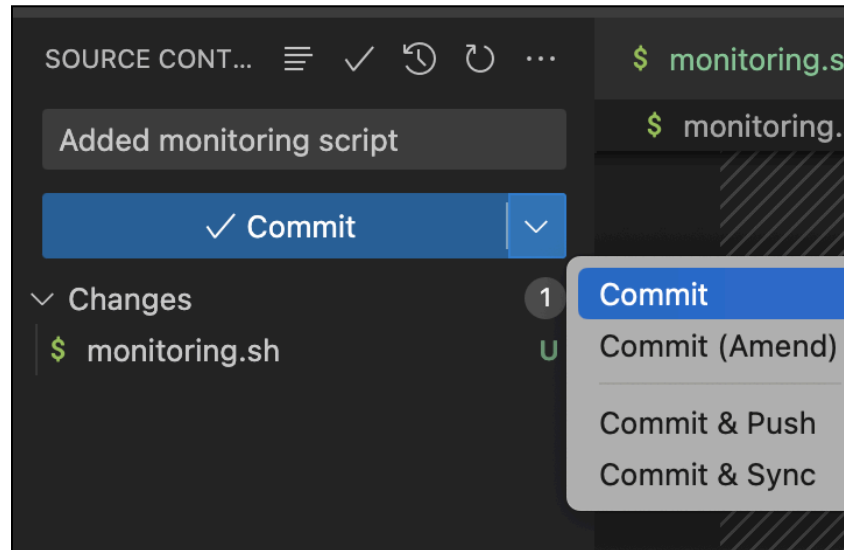


```
EXPLORER    ...    $ monitoring.sh U X
BASHSCRIPTS
> .idea
> .gitignore
> .zshrc
> 1000HelloWorld.sh
> aliases
E Amazon_Linux_LAMP_MediaWIKI
E Amazon_Linux_LAMP_Wordpress
E backup and restore MySQL DB use...
> backup_binlogs.sh
> connect_to_bastion.sh
> DBBackup.sh
DBCompare.sql
> DBRestore.sh
> disk_usage.sh
> dumpall.sh
> EBSmount.sh
- FeatureDevelopmentWithGit.pdf
> file_sort.sh
> GAC_GPOM.sh
E k8s_commands.txt
E Lightsail_Amazon_Linux_LAMP_W...
> loadtest.sh
> logrotate.sh
> monitoring.sh    U
$ monitoring.sh > ...
1  #!/bin/bash
2
3  # Версия операционной системы
4  os_version=$(cat /etc/os-release | grep "PRETTY_NAME" | cut -d '"' -f 2)
5
6  # Дата и время
7  current_date=$(date "+%Y-%m-%d")
8  current_time=$(date "+%H:%M:%S")
9
10 # Время работы системы
11 uptime_info=$(uptime -p)
12
13 # Загруженность системы
14 system_load=$(uptime | awk -F'[a-z]:' '{ print $2 }')
15
16 # Занятое дисковое пространство
17 disk_usage=$(df / | awk '{print $5}' | sed 's/%//')
18
19 # Топ процессы по использованию памяти
20 top_processes=$(ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -n 6)
21
22 # Количество процессов
23 process_count=$(ps -ef | wc -l)
24
25 # Количество пользователей
26 user_count=$(who | wc -l)
27
28 # Выводим отчет
29 echo "Отчет по системе"
30 echo "Версия операционной системы: $os_version"
```

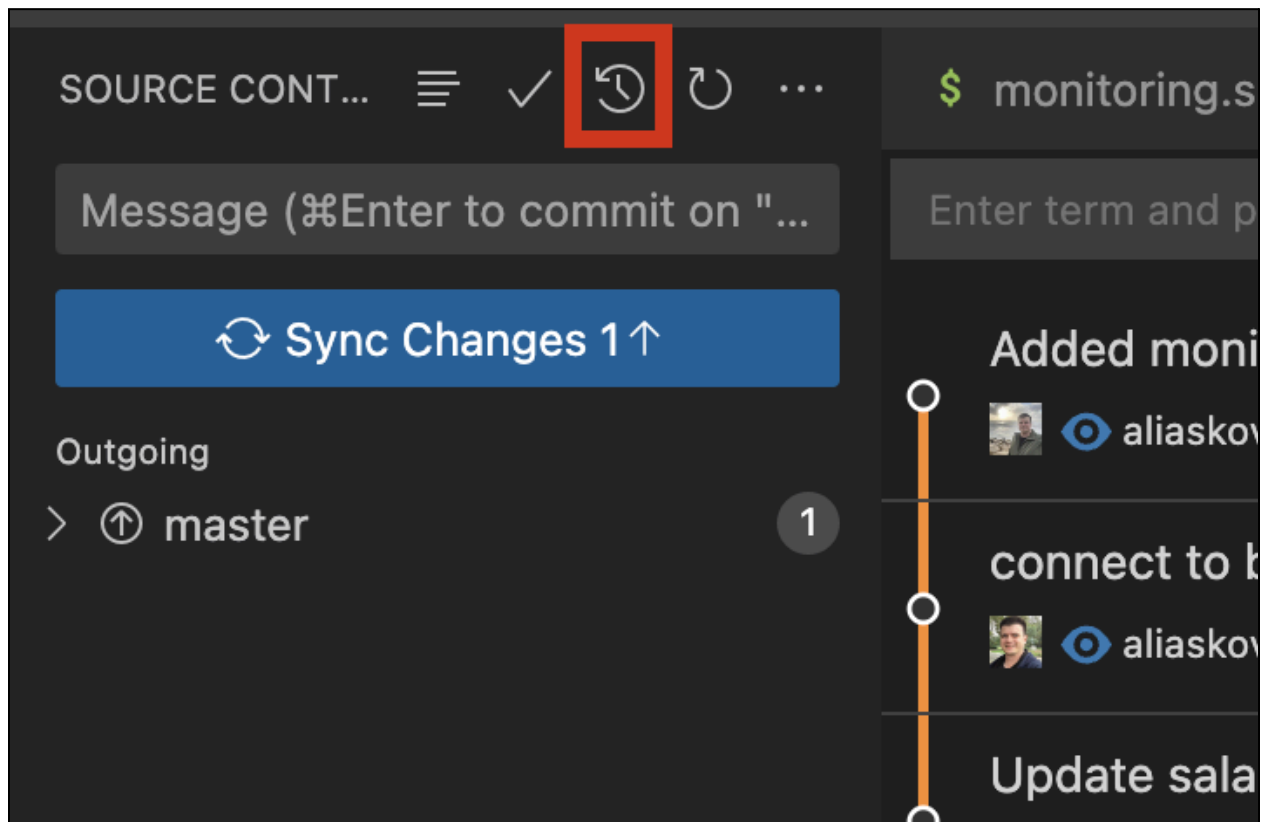
Не забываем сохранить его! Видим, что файл стал зеленым. Добавляем в отслеживание git. Нажимаем слева на иконку source control:



В удобном виде наблюдаем файлы с изменениями и можем оставить комментарий к коммиту (аналогичные команды в терминале это `git add` и `git commit -m`). Вносим сообщение к коммиту и нажимаем на Commit:

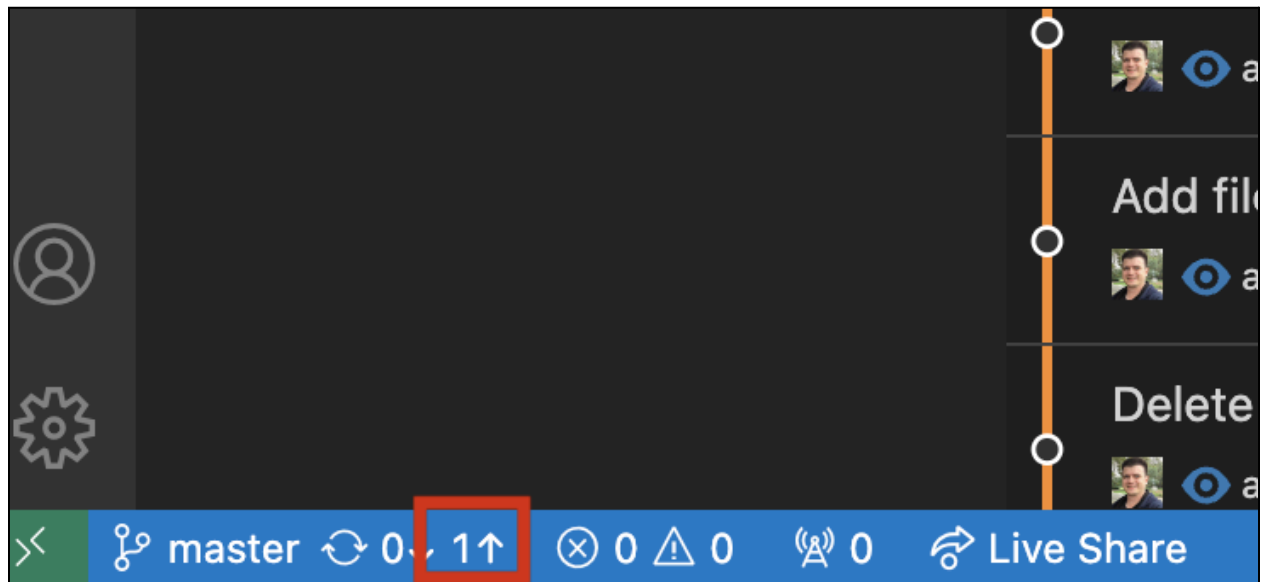


Обращаем внимание, что изменения закоммитились, можем посмотреть на историю коммитов:

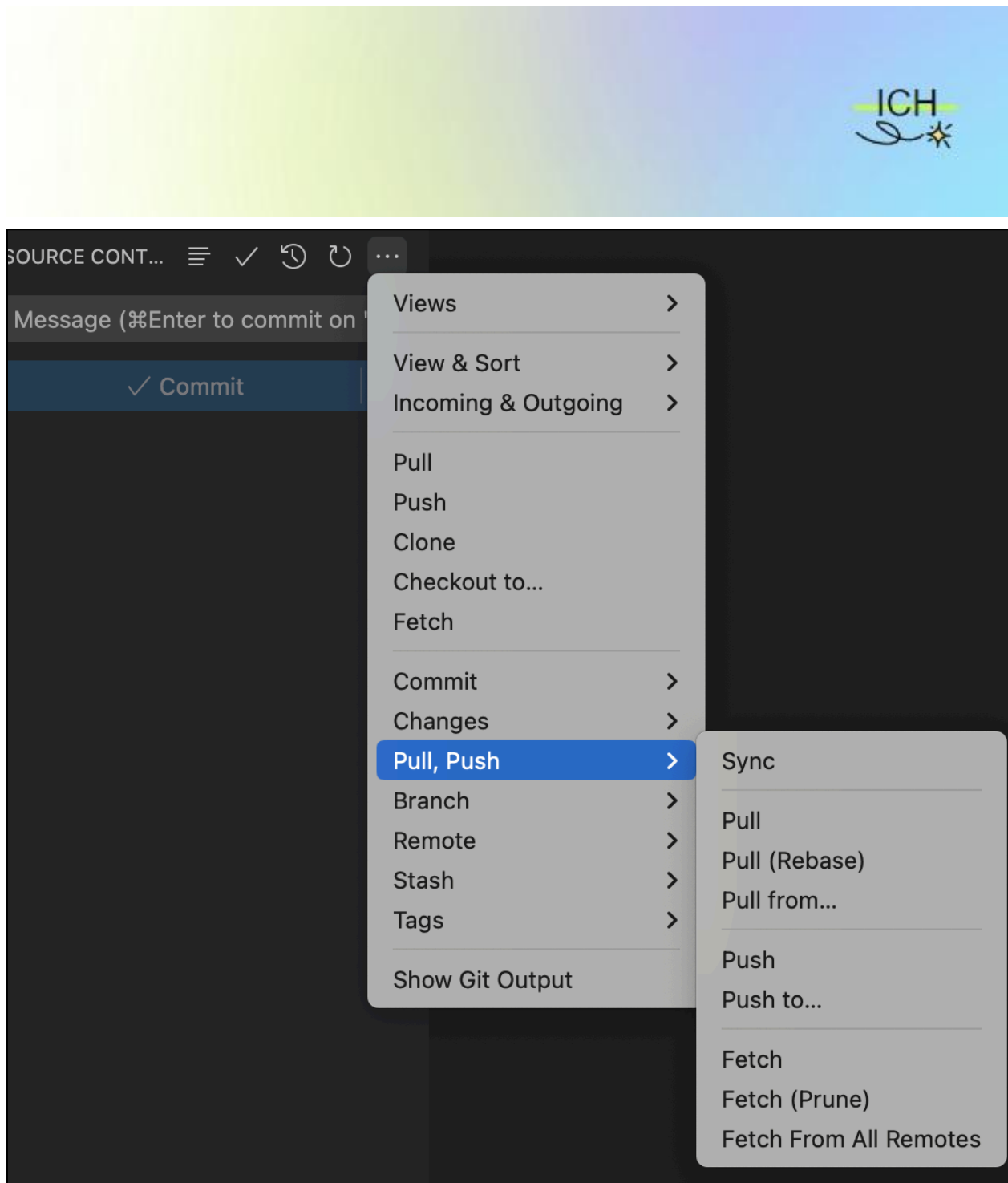




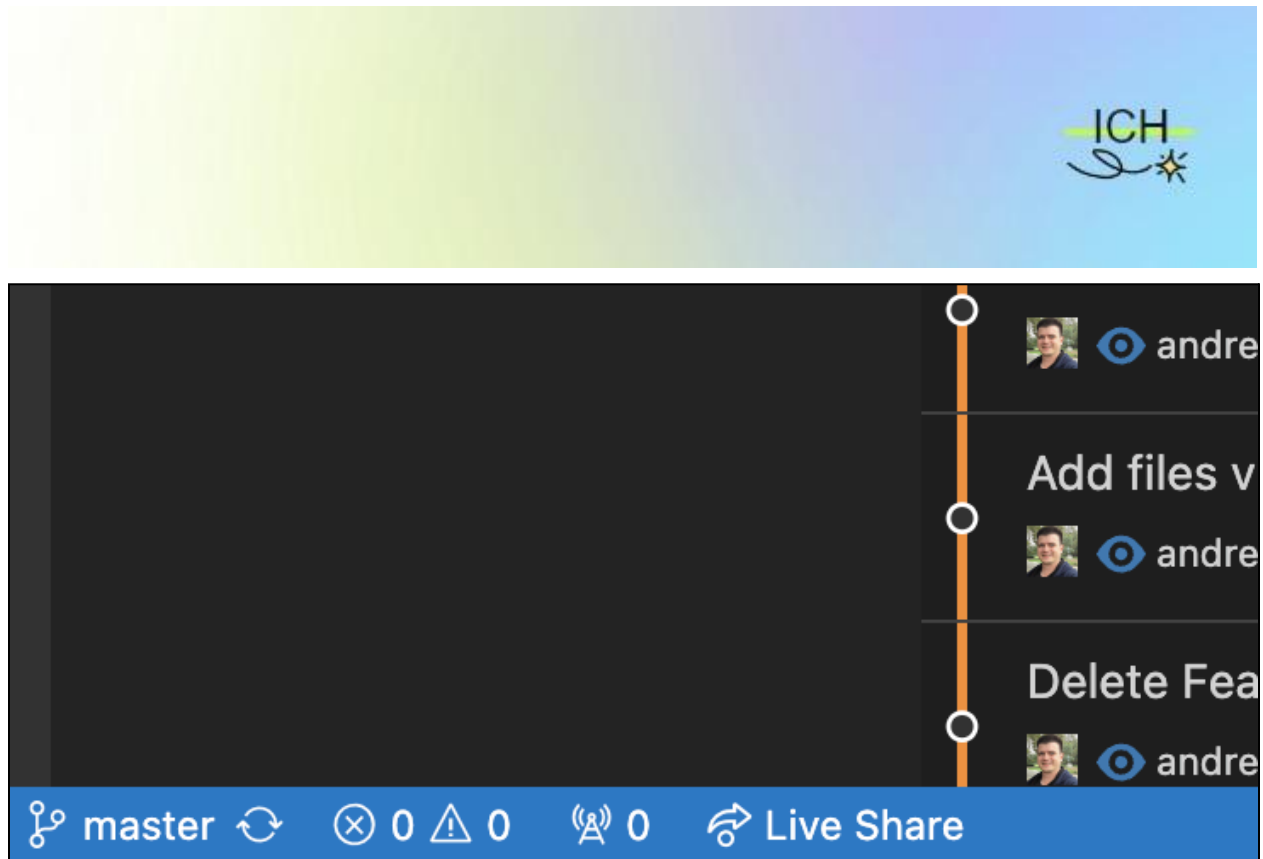
Пушим изменения. Обращаем внимание, что после нашего коммита внизу появилась индикация, что у нас есть коммиты, которые можно отправить в удалённый репозиторий:



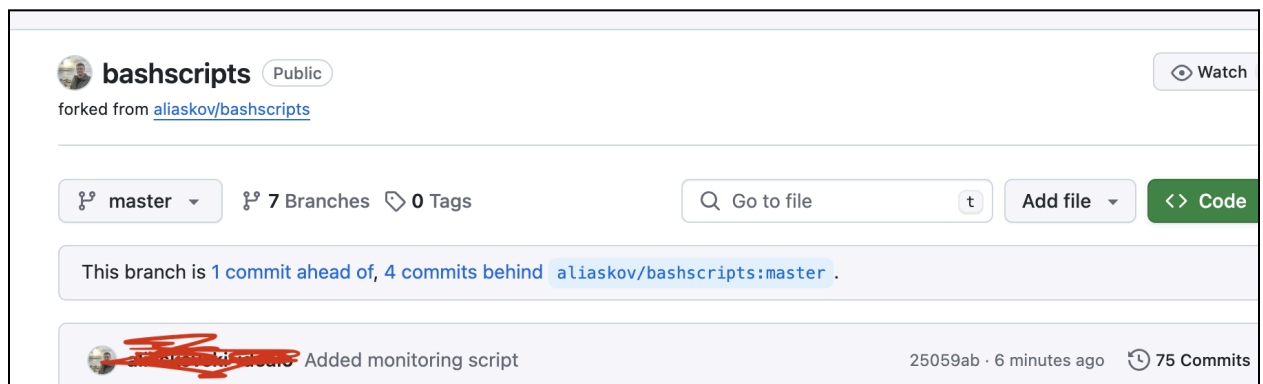
Жмем на эту кнопку и отправляем изменения. Другой вариант - через расширенное меню:



После успешной отправки - счетчики опять будут на нуле.



Проверяем в github, что новый скрипт появился в репозитории:





РАБОТА С PULL REQUEST

В Github открываем вкладку с Pull requests и нажимаем New pull request.
Выбираем направление для Pull request - из форкнутого репозитория в исходный и, если нас все устраивает, открываем 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](#) or [learn more about diff comparisons](#).

base repository: aliaskov/bashscripts base: master ← head repository: aliaskov/bashscripts compare: master

Source **Fork**

✓ **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

Добавляем основание для открытия PR , чаще всего это номер Issue, список которых можно получить через #:

Add a title
Added monitoring script

Add a description
Write Preview

#

- #50 Test
- #6 test deleted from gitignore
- #49 test
- #48 You must see it

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

После обсуждения и проверок Владелец исходного репозитория принимает изменения - делаем rebase and merge.



aliaskov / bashscripts

Issues 5

Pull requests 36

Discussions

Actions

Projects

Wiki

Added monitoring script #51

Merged


aliaskov merged 1 commit into `aliaskov:master` from `aliaskovsk`

Conversation 0

Commits 1


Checks 0


Files ch





aliaskov commented 2 minutes ago


[#48](#)





 Added monitoring script



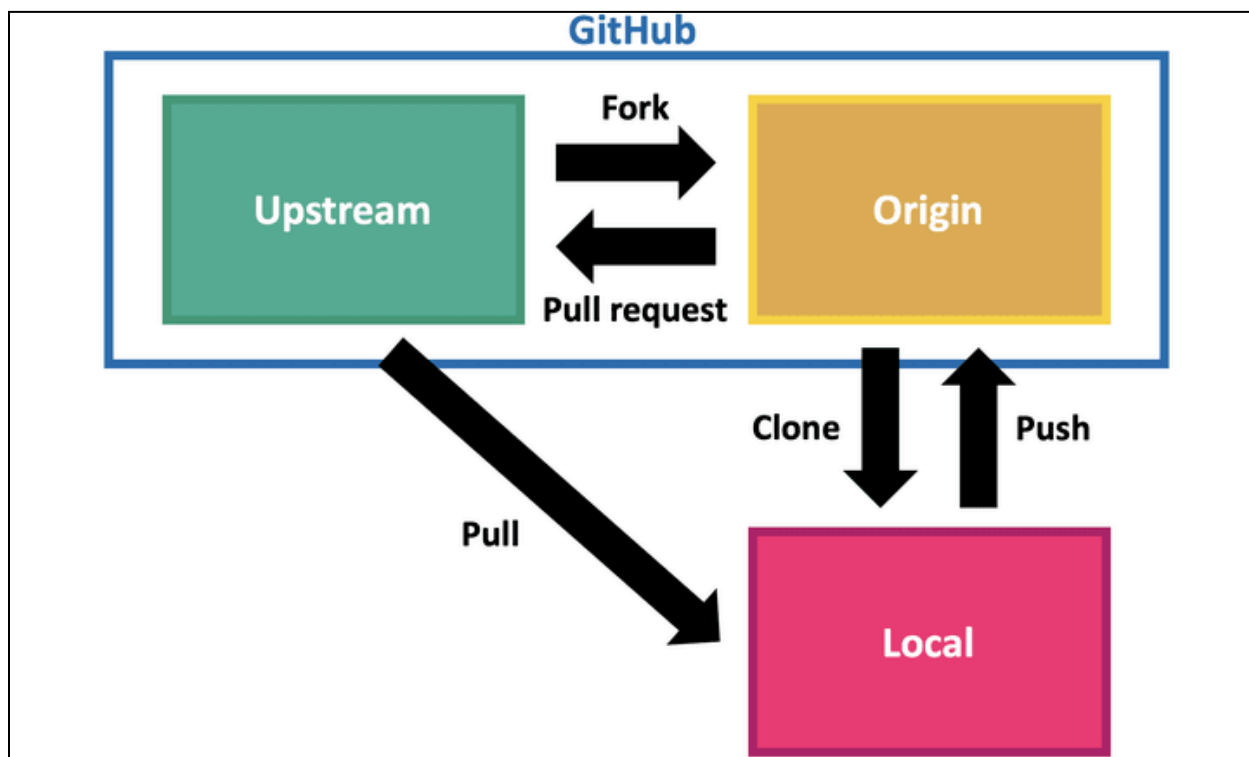
 aliaskov merged commit **8627aa2** into `aliaskov:master` 1 minute



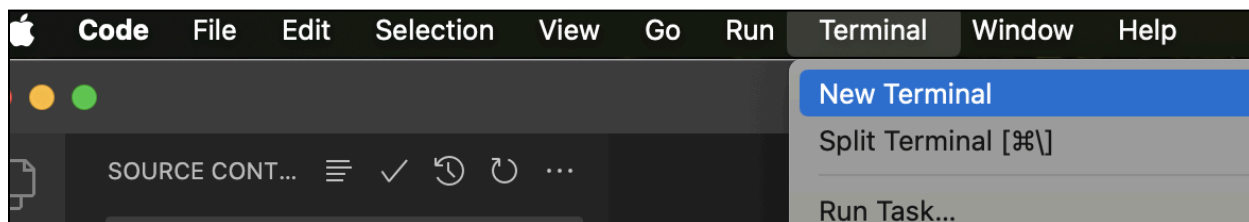
ОБНОВЛЕНИЕ ФОРКНУТОГО РЕПОЗИТОРИЯ

После того, как репозиторий принял изменения и замержил предложенную ветку, нам необходимо понять, как обновить репозиторий, который был форкнут.

Вот схема этого “треугольника”:



Чтобы обновить форкнутый репозиторий в Git:
Для удобства использовать терминал.





1. Добавить удаленный репозиторий оригинала (Обычно оригинальный репозиторий называется "upstream"):

Unset

```
git remote add upstream <URL_оригинального_репозитория (преподавателя)>
```

2. Получить изменения из оригинального репозитория (с помощью команды git fetch):

Unset

```
git fetch upstream
```

```
└─ git fetch upstream
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 3), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (12/12), 5.15 KiB | 405.00 KiB/s, done.
From https://github.com/aliaskov/bashscripts
* [new branch]      F2      -> upstream/F2
* [new branch]      F3      -> upstream/F3
* [new branch]      HW5     -> upstream/HW5
* [new branch]      master  -> upstream/master
```

3. Обновить вашу локальную ветку мастера (или другую ветку):

Unset

```
git checkout master
git merge upstream/master
```

```
└─ git merge upstream/master
Merge made by the 'ort' strategy.
 DBBackup.sh | 6 ++++++
 dumpall.sh  | 1 +
 salary.txt  | 1 +
 3 files changed, 8 insertions(+)
```

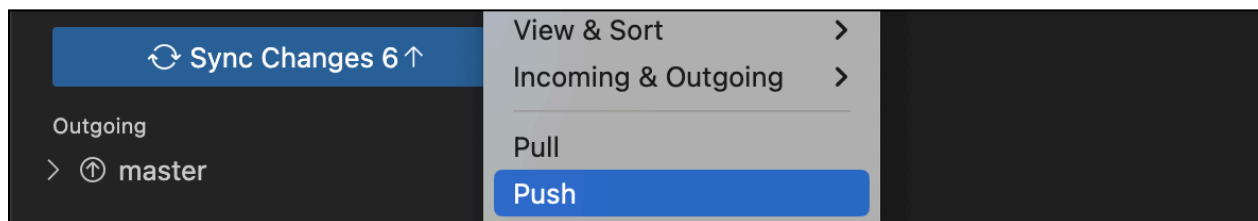


4. Зафиксировать и отправить изменения: если все прошло успешно и не возникло конфликтов, зафиксируйте и отправьте обновления в ваш форкнутый репозиторий, как обычно.

Unset

```
git push origin master
```

Или:



master может быть любой другой веткой (main или другая).