

Лабораторная работа 1.2

Тема: Исследование возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

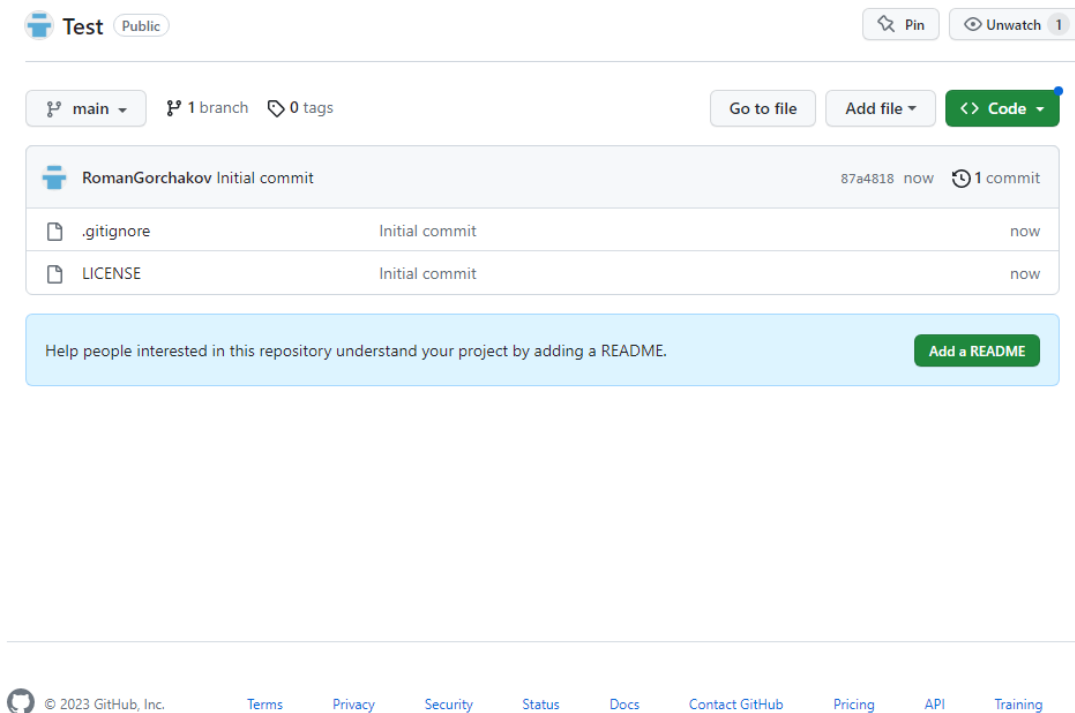
Choose a license

License: MIT License ▾

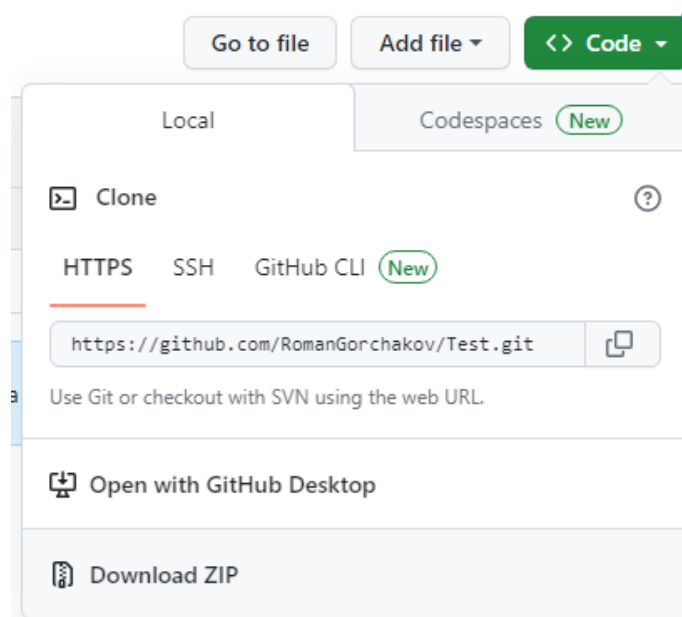
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.


















Create repository



2. В созданном репозитории нажимаем на зелёную кнопку Code, чтобы скопировать репозиторий на компьютер.



3. Теперь необходимо дополнить файл .gitignore необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

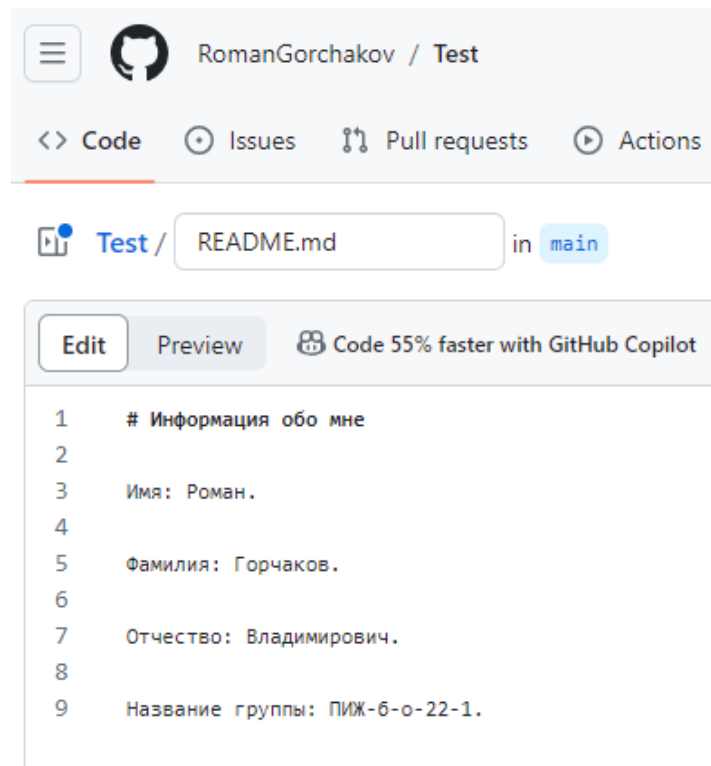
 Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
 PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
 Plone.gitignore	Covered by global vim template	10 years ago
 Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
 Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
 PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
 Python.gitignore	Update Python.gitignore	last year
 Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
 Qt.gitignore	Remove trailing whitespace	2 years ago
 R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
 README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
 ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
 Racket.gitignore	Update Racket.gitignore	2 years ago
 Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
 Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
 RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
 Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```

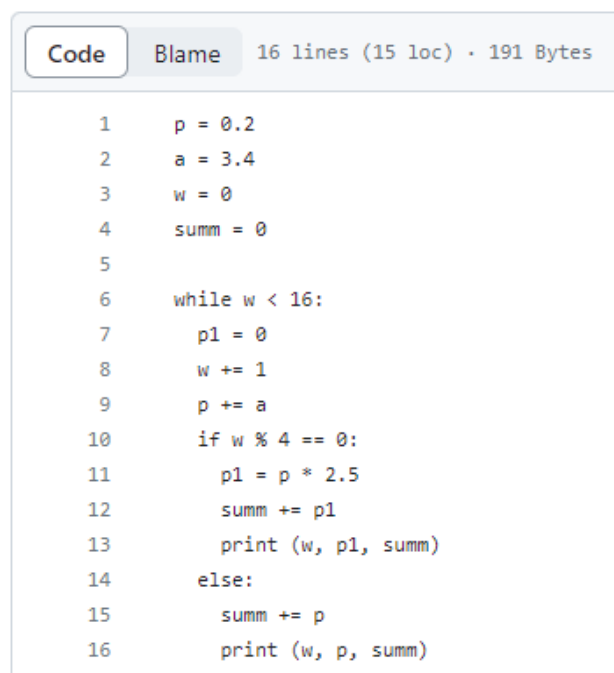
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28

```

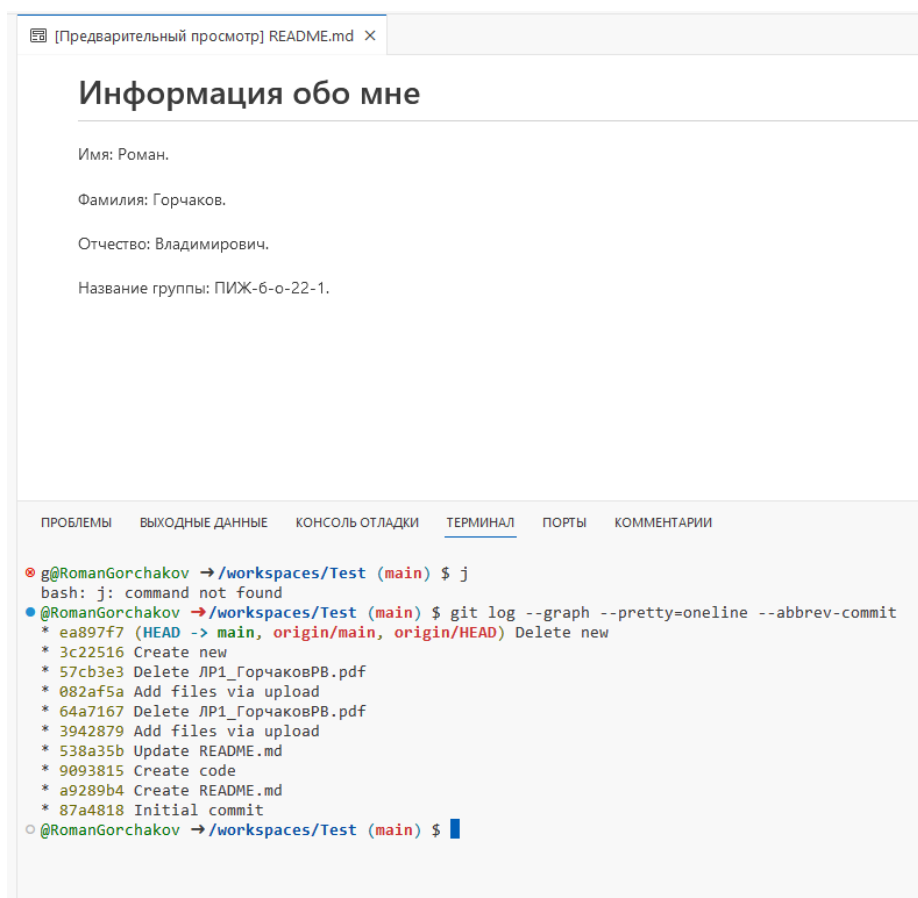
4. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».



5. Создаём новый файл «code», в которую запишем код для программы. Сохраняем изменения с помощью кнопки «Commit changes».



6. В окне «Codespace» выбираем опцию «Create codespace on main», где вводим команду «git log –graph –pretty=oneline –abbrev-commit».



The screenshot shows a GitHub Codespace environment. At the top, there's a tab for 'README.md'. Below it, the 'Информация обо мне' (About me) section is visible, containing personal details: 'Имя: Роман.', 'Фамилия: Горчаков.', 'Отчество: Владимирович.', and 'Название группы: ПИЖ-6-о-22-1.'. The bottom part of the interface features a terminal window with tabs for 'ПРОБЛЕМЫ', 'ВЫХОДНЫЕ ДАННЫЕ', 'КОНСОЛЬ ОТЛАДКИ', 'ТЕРМИНАЛ', 'ПОРТЫ', and 'КОММЕНТАРИИ'. The 'ТЕРМИНАЛ' tab is active, displaying the following commands and output:

```
@g@RomanGorchakov → /workspaces/Test (main) $ j
bash: j: command not found
@g@RomanGorchakov → /workspaces/Test (main) $ git log --graph --pretty=oneline --abbrev-commit
* ea897f7 (HEAD -> main, origin/main, origin/HEAD) Delete new
* 3c22516 Create new
* 57cb3e3 Delete ЛР1_ГорчаковРВ.pdf
* 082af5a Add files via upload
* 64a7167 Delete ЛР1_ГорчаковРВ.pdf
* 3942879 Add files via upload
* 538a35b Update README.md
* 9093815 Create code
* a9289b4 Create README.md
* 87a4818 Initial commit
@g@RomanGorchakov → /workspaces/Test (main) $
```

7. Просмотрим содержимое коммитов командами «git show HEAD», «git show HEAD~1» и «git show b34a0e».

```
[Предварительный просмотр] README.md X

Информация обо мне

Имя: Роман.

Фамилия: Горчаков.

Отчество: Владимирович.

Название группы: ПИЖ-6-о-22-1.

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  КОММЕНТАРИИ

● @RomanGorchakov → /workspaces/Test (main) $ git show HEAD
commit ea897f7d25d3f8d2d1c3ace647e05016534d200b (HEAD -> main, origin/main, origin/HEAD)
Author: RomanGorchakov <144220160+RomanGorchakov@users.noreply.github.com>
Date: Tue Sep 12 22:08:56 2023 +0300

Delete new

diff --git a/new b/new
deleted file mode 100644
index 8b13789..0000000
--- a/new
+++ /dev/null
@@ -1,0 @@

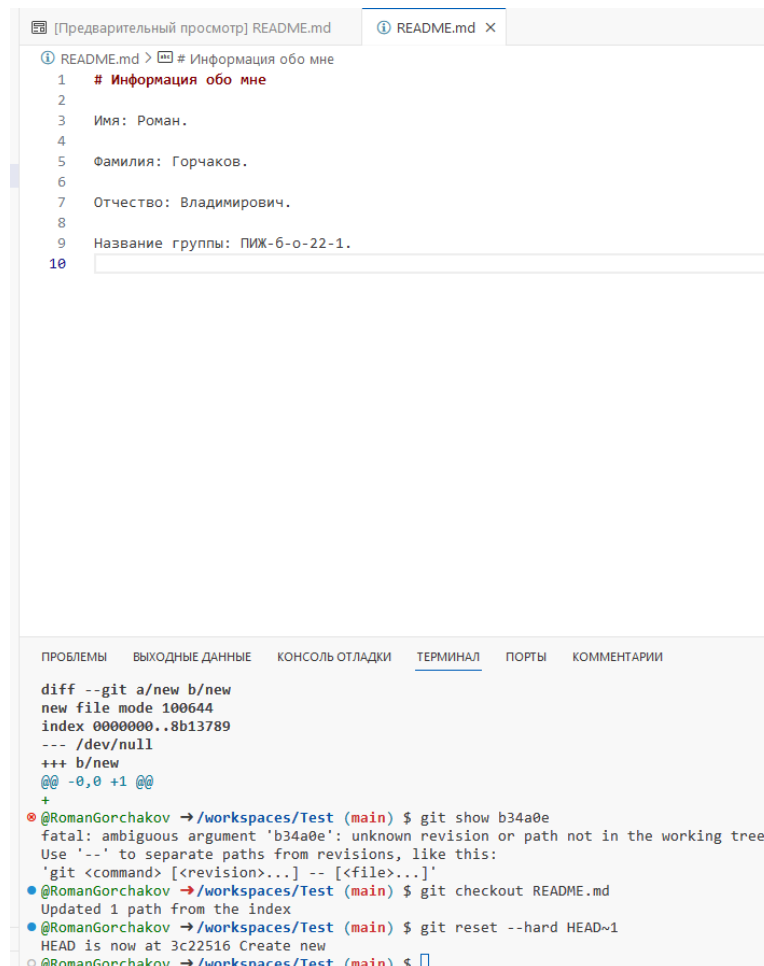
● @RomanGorchakov → /workspaces/Test (main) $ git show HEAD~1
commit 3c225160dabbbee32708c9dd86c8bd4d7b3c07b
Author: RomanGorchakov <144220160+RomanGorchakov@users.noreply.github.com>
Date: Tue Sep 12 22:06:45 2023 +0300

Create new

diff --git a/new b/new
new file mode 100644
index 0000000..8b13789
--- /dev/null
+++ b/new
@@ -0,0 +1 @@
+

● @RomanGorchakov → /workspaces/Test (main) $ git show b34a0e
fatal: ambiguous argument 'b34a0e': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
○ @RomanGorchakov → /workspaces/Test (main) $
```

8. Удаляем код из одного из файлов программы репозитория, после чего удалить все несохранённые изменения в файле командой «git checkout». Снова удалить код из того же файла, после чего откатить состояние хранилища к предыдущей версии командой «git reset».



The screenshot shows a code editor with two tabs: "[Предварительный просмотр] README.md" and "README.md". The README.md file contains the following text:

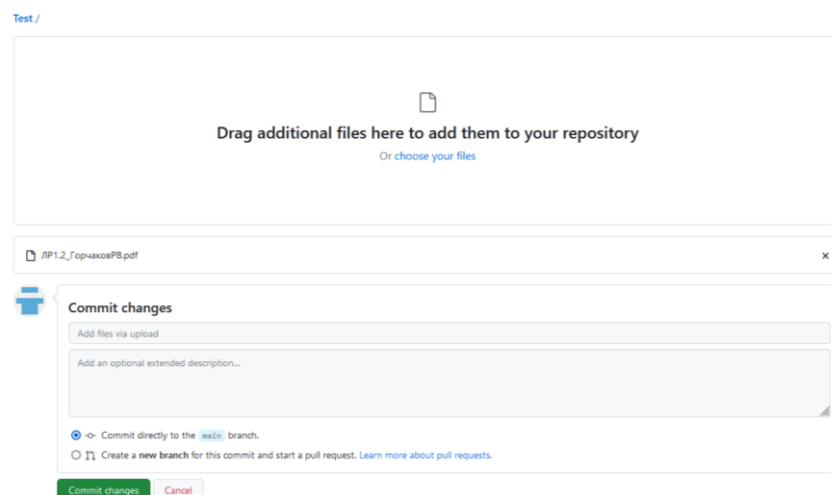
```
1 # Информация обо мне
2
3 Имя: Роман.
4
5 Фамилия: Горчаков.
6
7 Отчество: Владимирович.
8
9 Название группы: ПИЖ-6-о-22-1.
10
```

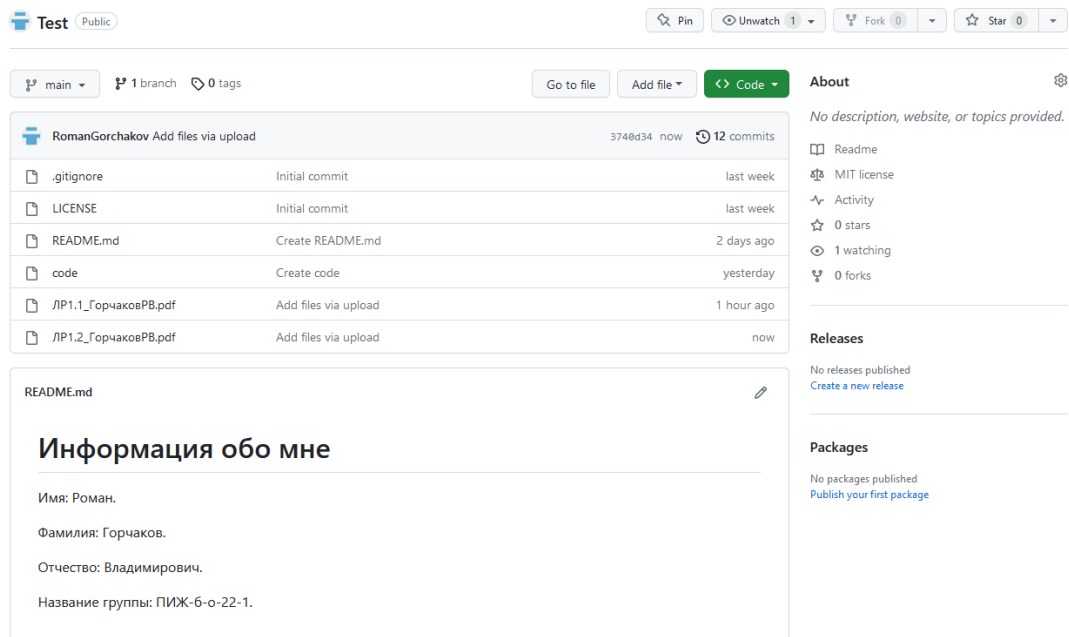
Below the editor is a terminal window with the following output:

```
ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ КОММЕНТАРИИ

diff --git a/new b/new
new file mode 100644
index 0000000..8b13789
--- /dev/null
+++ b/new
@@ -0,0 +1 @@
+
@RomanGorchakov → /workspaces/Test (main) $ git show b34a0e
fatal: ambiguous argument 'b34a0e': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
@RomanGorchakov → /workspaces/Test (main) $ git checkout README.md
Updated 1 path from the index
@RomanGorchakov → /workspaces/Test (main) $ git reset --hard HEAD~1
HEAD is now at 3c22516 Create new
@RomanGorchakov → /workspaces/Test (main) $
```

9. Сохраняем отчёт по лабораторной работе в качестве PDF-файла. Заходим в репозиторий на GitHub и нажимаем на кнопку «Add file». Выбираем опцию «Upload files», вставляем отчёт по лабораторной работе и нажимаем на кнопку «Commit changes».





Контрольные вопросы

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Историю коммитов в Git можно выполнить с помощью команды «git log».

Для просмотра истории коммитов используют опции «--patch», «--stat», «--pretty» и «--graph»

2. Как ограничить вывод при просмотре истории коммитов?

Ограничить вывод при просмотре истории коммитов можно с помощью опций «--since» и «--until»

3. Как внести изменения в уже сделанный коммит?

Внести изменения в уже сделанный коммит можно с помощью команды «git commit --amend».

4. Как отменить индексацию файла в Git?

Отменить индексацию файла в Git можно с помощью команды «git reset».

5. Как отменить изменения в файле?

Отменить изменения в файле можно с помощью команды «git checkout».

6. Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии проекта, сохранённые в интернете или ещё где-то в сети.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Просмотр удалённых репозиториях данного локального репозитория можно выполнить с помощью команды «git remote».

8. Как добавить удалённый репозиторий для данного локального репозитория?

Добавить удалённый репозиторий для данного локального репозитория можно с помощью команды «git remote add».

9. Как выполнить отправку/получение изменений с удалённого репозитория?

Выполнить отправку/получение изменений с удалённого репозитория можно с помощью команд «git push» и «git fetch»/«git pull».

10. Как выполнить просмотр удаленного репозитория?

Просмотр удалённого репозитория можно выполнить с помощью команды «git remote show».

11. Каково назначение тэгов Git?

Тэги в Git позволяют пометить определённые моменты в истории как важные.

12. Как осуществляется работа с тэгами Git?

Работа с тэгами Git осуществляется через команду «git tag».

13. Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага?

Команды «git fetch --prune» и «git push --prune» используются для очистки устаревших веток. Они подключаются к общему сетевому репозиторию и извлекают оттуда все ссылки на remote-ветки. Затем она удаляет remote-ссылки, которые больше не используются в remote-репозитории.