



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

**Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (МОСИТ)**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Технология разработки программных приложений»

Практическое занятие № 1

Студент группы *ИНБО-08-22 Самойлов М.М.*

(подпись)

Преподаватель *Мельников Д. А.*

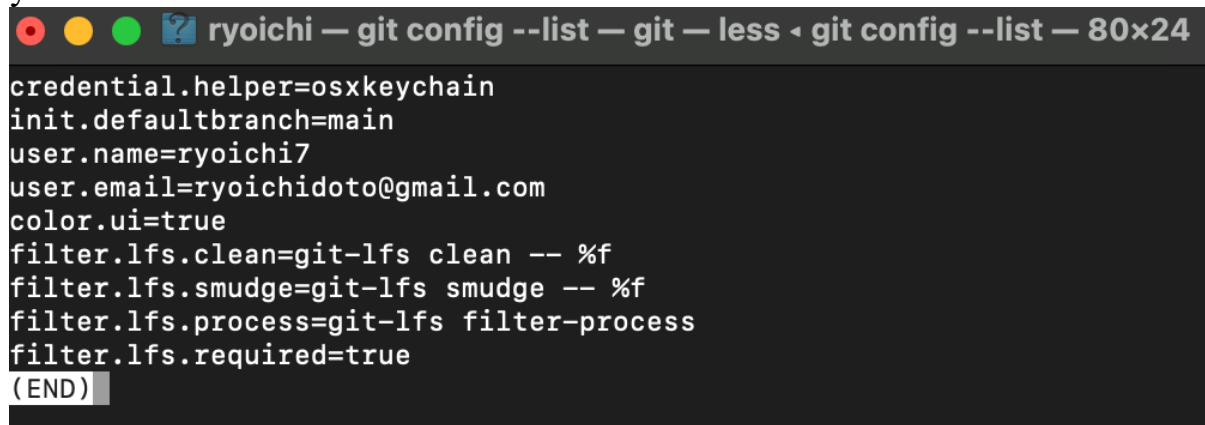
(подпись)

Отчет представлен «__» _____ 2024г.

Москва 2024 г.

1.1 Основные команды Git

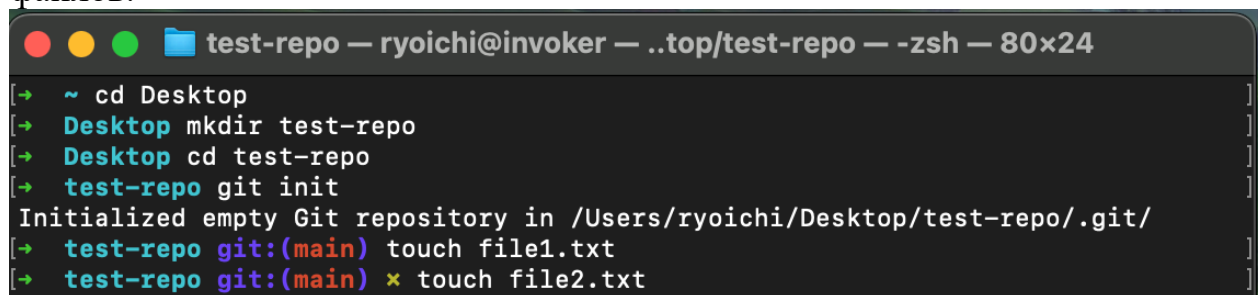
На ПК, в котором выполнялась практическая работа, git был уже установлен.

A terminal window titled 'ryoichi — git config --list — git — less ◀ git config --list — 80x24'. It displays the output of the 'git config --list' command, showing various configuration settings for the user 'ryoichi7'. The settings include credential.helper, init.defaultbranch, user.name, user.email, color.ui, and filter.lfs options. The terminal ends with '(END)' in a grey box.

```
credential.helper=osxkeychain
init.defaultbranch=main
user.name=ryoichi7
user.email=ryoichidoto@gmail.com
color.ui=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
(END)
```

Рисунок 1 – Конфигурация Git

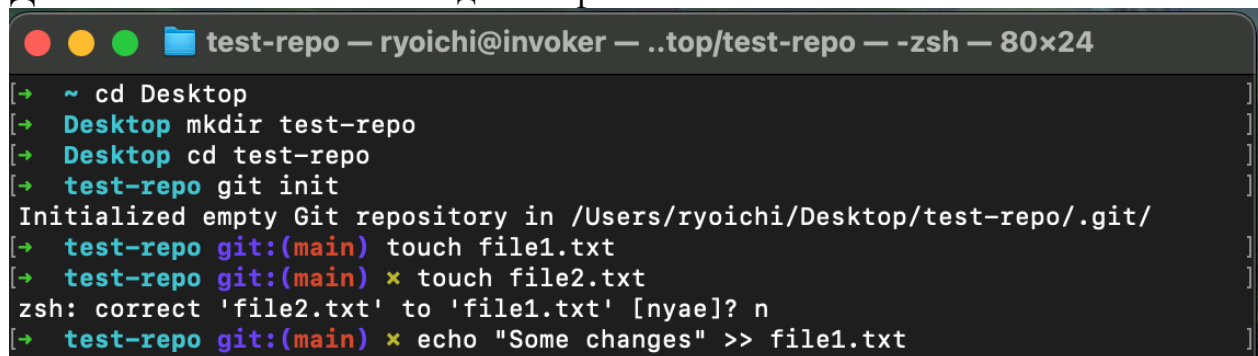
Далее создадим локальный репозиторий и добавим в него несколько файлов.

A terminal window titled 'test-repo — ryoichi@invoker — ../top/test-repo — -zsh — 80x24'. It shows a sequence of commands to create a local Git repository and add files. The commands are: 'cd Desktop', 'mkdir test-repo', 'cd test-repo', 'git init', 'git:(main) touch file1.txt', and 'git:(main) touch file2.txt'. The output shows the repository being initialized and the files being created.

```
~ cd Desktop
Desktop mkdir test-repo
Desktop cd test-repo
test-repo git init
Initialized empty Git repository in /Users/ryoichi/Desktop/test-repo/.git/
test-repo git:(main) touch file1.txt
test-repo git:(main) touch file2.txt
```

Рисунок 2 – создание локального репозитория и добавление файлов

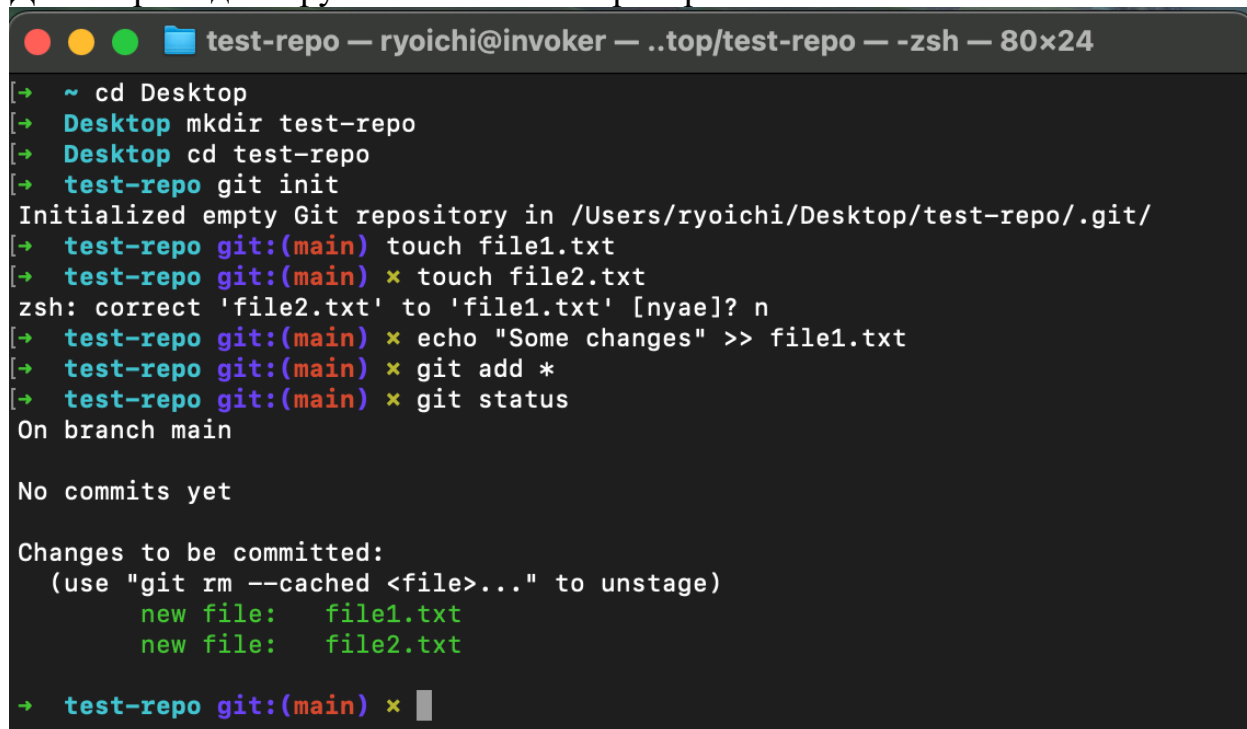
Далее внесем изменения в один из файлов.

A terminal window titled 'test-repo — ryoichi@invoker — ../top/test-repo — -zsh — 80x24'. It shows the same sequence of commands as in Figure 2, followed by an attempt to add 'file2.txt' which fails with a message from zsh suggesting it might be 'file1.txt'. The user then uses 'echo' to add content to 'file1.txt'.

```
~ cd Desktop
Desktop mkdir test-repo
Desktop cd test-repo
test-repo git init
Initialized empty Git repository in /Users/ryoichi/Desktop/test-repo/.git/
test-repo git:(main) touch file1.txt
test-repo git:(main) touch file2.txt
zsh: correct 'file2.txt' to 'file1.txt' [nyae]? n
test-repo git:(main) echo "Some changes" >> file1.txt
```

Рисунок 3 – внесение изменений в один из файлов

Далее проиндексируем изменения и проверим состояние.

A terminal window titled "test-repo — ryoichi@invoker — ../top/test-repo — zsh — 80x24". The terminal shows the following commands and output:

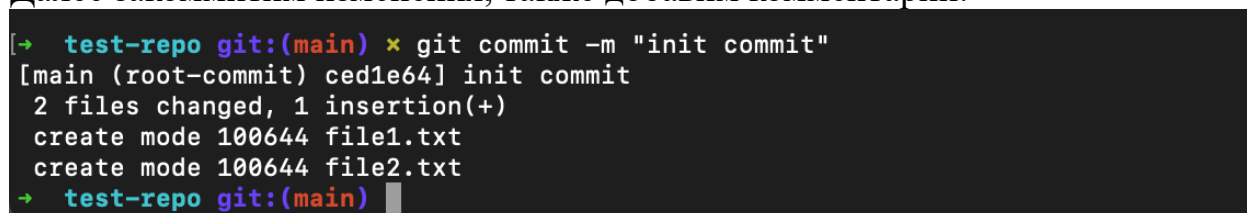
```
[→ ~ cd Desktop
[→ Desktop mkdir test-repo
[→ Desktop cd test-repo
[→ test-repo git init
Initialized empty Git repository in /Users/ryoichi/Desktop/test-repo/.git/
[→ test-repo git:(main) touch file1.txt
[→ test-repo git:(main) x touch file2.txt
zsh: correct 'file2.txt' to 'file1.txt' [nyae]? n
[→ test-repo git:(main) x echo "Some changes" >> file1.txt
[→ test-repo git:(main) x git add *
[→ test-repo git:(main) x git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
        new file:   file2.txt
[→ test-repo git:(main) x █
```

Рисунок 4 – индексация и проверка состояния

Далее закоммитим изменения, также добавим комментарии.

A terminal window showing the execution of a git commit command. The output is as follows:

```
[→ test-repo git:(main) x git commit -m "init commit"
[main (root-commit) ced1e64] init commit
2 files changed, 1 insertion(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
[→ test-repo git:(main) █
```

Рисунок 5 – коммит изменений

Далее изменим еще один файл. Проиндексируем это изменение, потом еще раз поменяем файл. Проверим состояние и закоммитим изменение. Далее добавим второе изменение в индекс, затем еще раз проверим состояние, далее проведем коммит.

```

[→ test-repo git:(main) echo "Another changes" >> file2.txt
[→ test-repo git:(main) * git add *
[→ test-repo git:(main) * echo "Another changes part 2" >> file2.txt
[→ test-repo git:(main) * git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2.txt

[→ test-repo git:(main) * git commit -a -m "add changes"
[main 5af99ac] add changes
 1 file changed, 2 insertions(+)
[→ test-repo git:(main) echo "Fix some bugs in previous changes" >> file2.txt
[→ test-repo git:(main) * git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2.txt

no changes added to commit (use "git add" and/or "git commit -a")
[→ test-repo git:(main) * git commit -a -m "fix bugs in file2"
[main 9d11144] fix bugs in file2
 1 file changed, 1 insertion(+)
[→ test-repo git:(main) git status
On branch main
nothing to commit, working tree clean
→ test-repo git:(main) █

```

Рисунок 6 – проверка статуса в разных ситуациях

Просмотрим историю коммитов с помощью команды `git log`. Ознакомимся с параметрами команды и используем их для различного формата изображения коммитов.

```
commit 9d11144adef5883d640761a1ada9a237b3a9e287 (HEAD -> main)
Author: ryoichi7 <ryoichidoto@gmail.com>
Date: Tue Feb 27 21:45:15 2024 +0300

    fix bugs in file2

commit 5af99aca16c8f377e195f16150a1b617d65c99f1
Author: ryoichi7 <ryoichidoto@gmail.com>
Date: Tue Feb 27 21:42:01 2024 +0300

    add changes

commit ced1e64d9a5ee6f86ee711721920af3f8b67b59f
Author: ryoichi7 <ryoichidoto@gmail.com>
Date: Tue Feb 27 21:39:54 2024 +0300

    init commit
(END)
```

Рисунок 7 – проверка истории коммитов

Вернем рабочий каталог к одному из предыдущих состояний.

```
[→ test-repo git:(main) git checkout HEAD^
Note: switching to 'HEAD^'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 5af99ac add changes
→ test-repo git:(5af99ac)
```

Рисунок 8 – возвращение каталога к одному из предыдущих состояний

Применим тег.

```
[→ test-repo git:(5af99ac) git tag -a v1.0 -m "Version 1.0"
→ test-repo git:(v1.0)
```

Рисунок 9 – применение тега

Отменим изменения в файле до индексации.

```
[→ test-repo git:(v1.0) git checkout -- file2.txt  
→ test-repo git:(v1.0) █
```

Рисунок 10 – отмена изменений до индексации

Отмена изменений после индексации.

```
[→ test-repo git:(v1.0) git reset HEAD file2.txt  
→ test-repo git:(v1.0) █
```

Рисунок 11 – отмена изменений после индексации

Отменим коммит.

```
[→ test-repo git:(v1.0) git revert head  
[detached HEAD a4b06b0] Revert "add changes"  
1 file changed, 2 deletions(-)  
→ test-repo git:(a4b06b0) █
```

Рисунок 12 – отмена коммита

1.2 Системы управления репозиториями

Аккаунт GitHub создан.

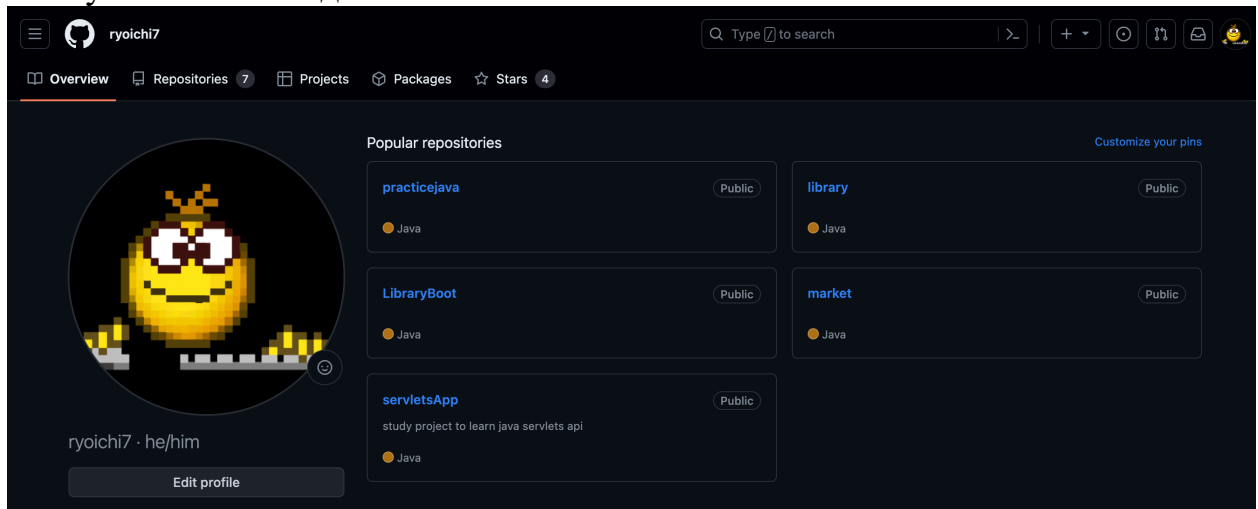


Рисунок 13 – Созданный аккаунт github

Создан пустой репозиторий на гитхабе.

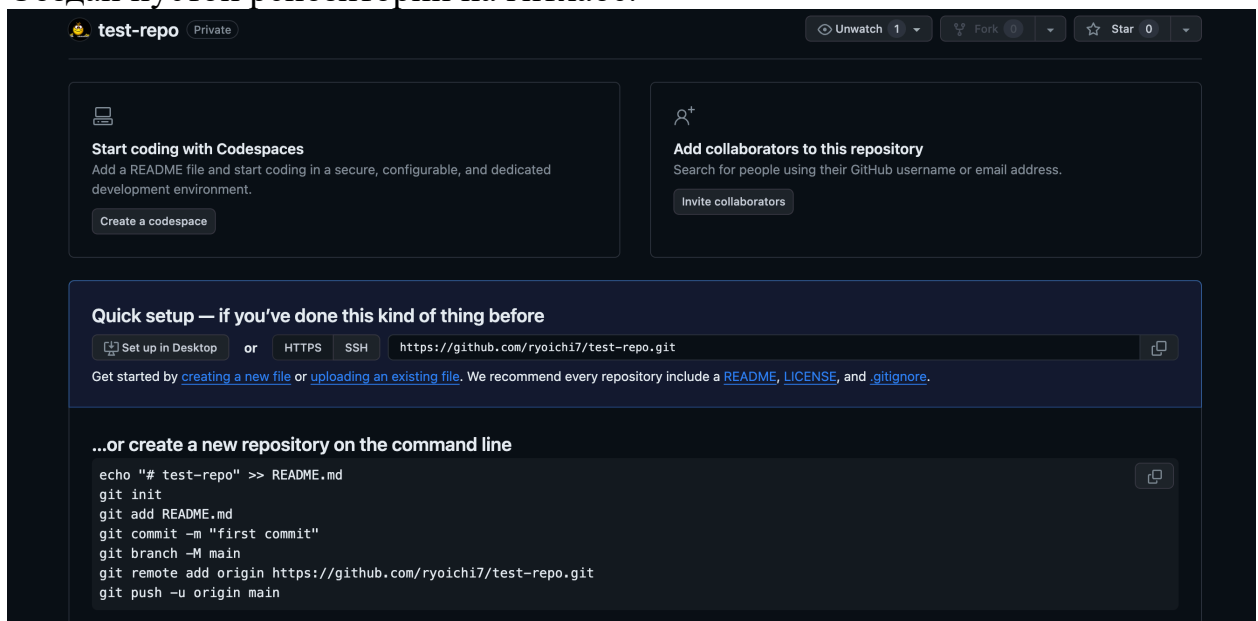


Рисунок 14 – Созданный репозиторий github

Создан пустой репозиторий на ПК.


```

[→ Desktop mkdir test-repo
[→ Desktop cd test-repo
[→ test-repo git init
Initialized empty Git repository in /Users/ryoichi/Desktop/test-repo/.git/
[→ test-repo git:(main) %

```

Рисунок 15 – Созданный локальный репозиторий

Добавлен коммит.

```

[→ test-repo git:(main) touch file1.txt
[→ test-repo git:(main) x echo "some data" >> file1.txt
[→ test-repo git:(main) x git add *
[→ test-repo git:(main) x git commit -m "init commit"
[main (root-commit) bdae663] init commit
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
[→ test-repo git:(main) █

```

Рисунок 16 – Добавление коммита

Локальный репозиторий подключен к репозиторию на гитхабе.

```

[→ test-repo git:(main) git remote add origin https://github.com/ryoichi7/test-repo.git
[→ test-repo git:(main) █

```

Рисунок 17 – Подключение локального репозитория к удаленному

Создана ветка, а потом слита с основной.

```

[→ test-repo git:(main) git branch dev
[→ test-repo git:(main) git checkout dev
Switched to branch 'dev'
[→ test-repo git:(dev) touch devtools.txt
[→ test-repo git:(dev) x echo "Unsafe changes" >> devtools.txt
[→ test-repo git:(dev) x git commit -m "unsafe changes"
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    devtools.txt

nothing added to commit but untracked files present (use "git add" to track)
[→ test-repo git:(dev) x git add *
[→ test-repo git:(dev) x git commit -m "unsafe changes"
[dev 149d8d5] unsafe changes
1 file changed, 1 insertion(+)
create mode 100644 devtools.txt
[→ test-repo git:(dev) git checkout main
Switched to branch 'main'
[→ test-repo git:(main) git merge dev
Updating bdae663..149d8d5
Fast-forward
 devtools.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 devtools.txt
[→ test-repo git:(main) █

```

Рисунок 18 – Слияние основной ветки с новой

Работа с отложенными изменениями - git stash.

```
[→ test-repo git:(main) echo "changes to be placed in stash" >> file1.txt
[→ test-repo git:(main) ✕ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
[→ test-repo git:(main) ✕ git stash
Saved working directory and index state WIP on main: 149d8d5 unsafe changes
[→ test-repo git:(main) git status
On branch main
nothing to commit, working tree clean
[→ test-repo git:(main) cat file1.txt
some data
[→ test-repo git:(main) git branch stashBranch
[→ test-repo git:(main) git checkout stashBranch
Switched to branch 'stashBranch'
[→ test-repo git:(stashBranch) git stash pop
On branch stashBranch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (61aae1f6b6713f155ab6c819e8c629b166f0411c)
[→ test-repo git:(stashBranch) ✕ git commit -a -m "add changes from stash"
[stashBranch 68052f3] add changes from stash
 1 file changed, 1 insertion(+)
→ test-repo git:(stashBranch) █
```

Рисунок 19 – Созданный аккаунт github

1.3 Работа с ветвлением и оформление кода

Сделан форк репозитория.

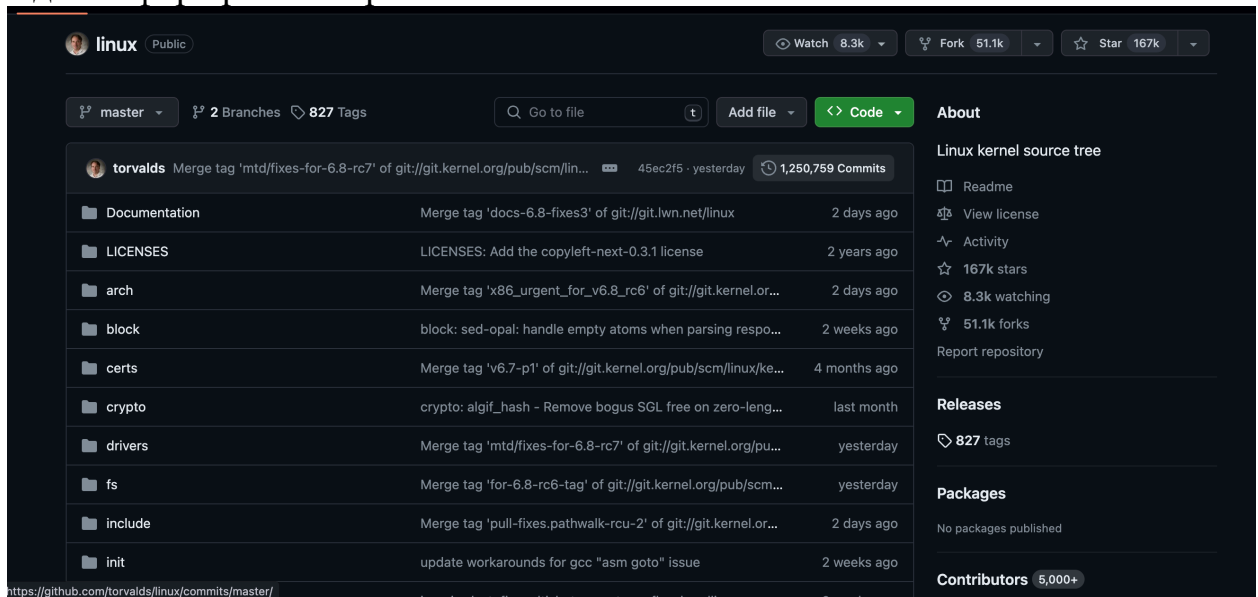


Рисунок 20 – форк репозитория

Склонирован репозиторий

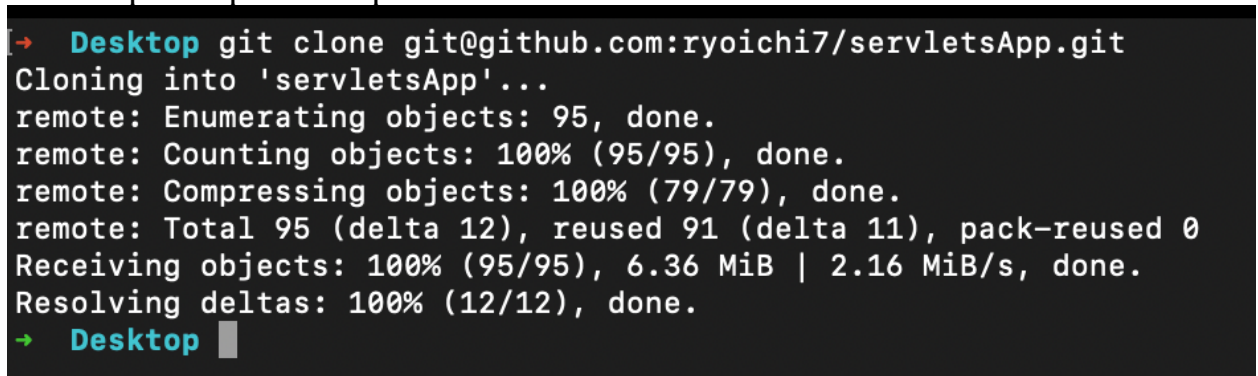


Рисунок 21 – клонирование репозитория

Создано 2 ветки.

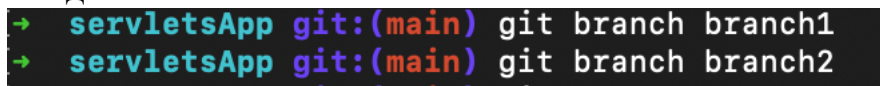


Рисунок 22 – создание 2 веток

Три коммита на первой ветке.

```
→ servletsApp git:(main) git checkout branch1
Switched to branch 'branch1'
→ servletsApp git:(branch1) echo "changes from branch1" >> file1.txt
→ servletsApp git:(branch1) * git add *
→ servletsApp git:(branch1) * git commit -m "change file from branch1"
[branch1 ff11fd0] change file from branch1
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
→ servletsApp git:(branch1) echo "another changes from branch1" >> file1.txt
→ servletsApp git:(branch1) * git add *
→ servletsApp git:(branch1) * git commit -m "change file 2nd time from branch1"
[branch1 c1b301b] change file 2nd time from branch1
1 file changed, 1 insertion(+)
→ servletsApp git:(branch1) echo "third changes from branch1" >> file1.txt
→ servletsApp git:(branch1) * git add *
→ servletsApp git:(branch1) * git commit -m "change file 3rd time from branch1"
[branch1 d2ccc79] change file 3rd time from branch1
1 file changed, 1 insertion(+)
→ servletsApp git:(branch1) █
```

Рисунок 23 – 3 коммита в ветке 1

Три коммита на второй ветке.

```
→ servletsApp git:(branch2) * echo "changes from branch2" >> file1.txt
→ servletsApp git:(branch2) * git add *
→ servletsApp git:(branch2) * git commit -m "change file from branch2"
[branch2 2c29562] change file from branch2
1 file changed, 2 insertions(+)
create mode 100644 file1.txt
→ servletsApp git:(branch2) echo "another changes from branch2" >> file1.txt
→ servletsApp git:(branch2) * git add *
→ servletsApp git:(branch2) * git commit -m "change file 2nd time from branch2"
[branch2 3c045ad] change file 2nd time from branch2
1 file changed, 1 insertion(+)
→ servletsApp git:(branch2) echo "third changes from branch2" >> file1.txt
→ servletsApp git:(branch2) * git add *
→ servletsApp git:(branch2) * git commit -m "change file 2nd time from branch2"
[branch2 b6eaa7d] change file 2nd time from branch2
1 file changed, 1 insertion(+)
→ servletsApp git:(branch2) █
```

Рисунок 24 – 3 коммита в ветке 2

Слияние двух веток с устранением конфликта.

```
changes from branch1
<<<<<< HEAD
another changes from branch1
third changes from branch1
=====
changes from branch2
another changes from branch2
third changes from branch2
>>>>>> branch2
```

Рисунок 25 – Конфликт веток

```
[→ servletsApp git:(branch1) git merge branch2
Auto-merging file1.txt
CONFLICT (add/add): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
[→ servletsApp git:(branch1) ✕ nano file1.txt
[→ servletsApp git:(branch1) ✕ git add *
[→ servletsApp git:(branch1) git commit -m "merge branch1 and branch2"
[branch1 2556f4c] merge branch1 and branch2
[→ servletsApp git:(branch1) git status
On branch branch1
nothing to commit, working tree clean
→ servletsApp git:(branch1)
```

Рисунок 26 – Слияние веток после решения конфликта

Коммиты в исходном репозитории

```
→ delete git:(main) echo "1st change" >> file1.txt
→ delete git:(main) ✕ git commit -a -m "1st change commit"
[main a367b3e] 1st change commit
1 file changed, 1 insertion(+)
→ delete git:(main) echo "2nd change" >> file1.txt
→ delete git:(main) ✕ git commit -a -m "2nd change commit"
[main a5c7dd5] 2nd change commit
1 file changed, 1 insertion(+)
→ delete git:(main) echo "3rd change" >> file1.txt
→ delete git:(main) ✕ git commit -a -m "3rd change commit"
[main 1edccc5] 3rd change commit
1 file changed, 1 insertion(+)
→ delete git:(main)
```

Рисунок 27 – 3 коммита

3 коммита в новом клоне репозитории.

```
[→ deleteNew git:(main) echo "1st change" >> file1.txt ]
[→ deleteNew git:(main) * git commit -a -m "1st change from cloned repo" ]
[main 019ed30] 1st change from cloned repo
1 file changed, 1 insertion(+)
[→ deleteNew git:(main) echo "2nd change" >> file1.txt ]
[→ deleteNew git:(main) * git commit -a -m "2nd change from cloned repo" ]
[main d47def3] 2nd change from cloned repo
1 file changed, 1 insertion(+)
[→ deleteNew git:(main) echo "3rd change" >> file1.txt ]
[→ deleteNew git:(main) * git commit -a -m "3rd change from cloned repo" ]
[main 1eb2cbc] 3rd change from cloned repo
1 file changed, 1 insertion(+)
```

Рисунок 28 – 3 коммита в новом клоне

Пуш нового клона в репозиторий.

```
[→ deleteNew git:(main) git push origin main ]
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 785 bytes | 785.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ryoichi7/delete.git
b875580..1eb2cbc main -> main
[→ deleteNew git:(main) ]
```

Рисунок 29 – пуш нового клона в удаленный репозиторий

Пуш начального локального репозитория в удаленный репозиторий.

```
[→ deleteNew git:(main) cd ../delete ]
[→ delete git:(main) git push origin main --force ]
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (12/12), 955 bytes | 955.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ryoichi7/delete.git
+ 1eb2cbc...1edccc5 main -> main (forced update)
[→ delete git:(main) ]
```

Рисунок 30– пуш начального локального репозитория с –force

Пул в новый клонированный репозиторий

```
[→ deleteNew git:(main) git pull git@github.com:ryoichi7/delete.git ]
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (12/12), 935 bytes | 93.00 KiB/s, done.
From github.com:ryoichi7/delete
 * branch                HEAD      -> FETCH_HEAD
→ deleteNew git:(main)
```

Рисунок 31— пул в клонированный репозиторий

Вывод

В ходе практической работы по Git и GitHub были изучены основные концепции и функционал этих инструментов для контроля версий и совместной работы над проектами. Был установлен и настроен Git на локальной машине, что позволило начать работу с системой контроля версий. Создали локальный репозиторий для хранения версий и истории проекта. Изучили и применили основные команды Git, такие как `git add`, `git commit`, `git push` и `git pull`, для управления изменениями, фиксации версий и синхронизации с удаленным репозиторием. Создали и научились использовать ветки, выполнили слияния изменений и разрешили конфликты, что позволило эффективно организовать работу над проектом в команде. Был создан удаленный репозиторий на платформе GitHub и связан с локальным репозиторием. Были выполнены операции по синхронизации изменений между локальным и удаленным репозиториями.

Вопросы:

Что такое репозиторий git?

Репозиторий Git - это хранилище данных, в котором хранятся файлы вашего проекта и история изменений этих файлов.

Что такое коммит?

Коммит (commit) в системе контроля версий Git - это операция, при которой изменения в файлах проекта сохраняются в репозитории.

Что такое ветка в репозитории git?

Ветка (branch) в репозитории Git представляет собой легковесную подвижную ссылку на один из коммитов в истории репозитория. Она позволяет разработчикам работать параллельно над различными версиями проекта, не затрагивая основную ветку разработки.

Что такое слияние веток?

Слияние двух веток в системе контроля версий Git - это процесс объединения изменений из одной ветки с другой. Когда вы сливаете две ветки, Git объединяет изменения, сделанные в обеих ветках, чтобы создать новый коммит, содержащий комбинацию изменений из обеих веток.

Для чего нужен gitignore?

Файл.gitignore в репозитории Git используется для указания файлов и каталогов, которые Git должен игнорировать при отслеживании изменений в репозитории

Что делает git status?

Команда git status используется для получения информации о текущем состоянии рабочего каталога и индекса в репозитории Git.

Что делает git add?

Команда git add используется для добавления изменений в файлы в индекс перед тем, как зафиксировать их в новом коммите.

Что делает git log?

Команда git log используется для просмотра истории коммитов в репозитории Git.