Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Работа: Игнатов М. В., M3112, Инструментальные средства разработки ПО, работа №2

Выполнил: Игнатов Максим Вячеславович

Проверил: Повышев Владислав Вячеславович

Санкт-Петербург

2022 г.

1. Создание репозитория и веток по модели Git Flow.

```
maksimignatov@MacBook-Air-Maksim-3 ~ % cd /Users/maksimignatov/Documents/Gitlab2
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git flow init
Initialized empty Git repository in /Users/maksimignatov/Documents/Gitlab2/.git/
```

```
How to name your supporting branch prefixes?
Feature branches? [feature/] feature/
Bugfix branches? [bugfix/] bugfix/
Release branches? [release/] release/
Hotfix branches? [hotfix/] hotfix/
Support branches? [support/] support/
Version tag prefix? []
Hooks and filters directory? [/Users/maksimignatov/Documents/Gitlab2/.git/hooks] /Users/maksimignatov/Documents/Gitlab2/.git/hooks
```

2. Создание и добавление в репозиторий README файла.

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % touch README.md
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % nano README.md
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % cat README.md
# Lab2
## Submodules in lab:
        geometric_lib
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git add README.md
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git commit -m "README.md was added"
[develop 6126f31] README.md was added
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
```

3. Создание ветки Feature, переключение на нее, а также создание первого тега.

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git flow feature start Feature_Maksim
Switched to a new branch 'feature/Feature_Maksim'

Summary of actions:
- A new branch 'feature/Feature_Maksim' was created, based on 'develop'
- You are now on branch 'feature/Feature_Maksim'

Now, start committing on your feature. When done, use:

     git flow feature finish Feature_Maksim
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git tag -a v1.0 -m "Start of developing"
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git show  v1.0
tag v1.0
Tagger: Maksim <ignatov.maksim.04@mail.ru>
Date:   Thu Sep 29 23:11:02 2022 +0300

Start of developing

commit 9cfbddc1f5ed2b09c6b8b83a671ab21b6d03625b (HEAD -> feature/Feature_Maksim, tag: v1.0, origin/develop, origin/HEAD, develop)
Author: Maksim <ignatov.maksim.04@mail.ru>
Date:   Thu Sep 29 14:45:37 2022 +0300

    Updated README.md

diff --git a/README.md b/README.md
index 9baf252..663fb2e 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,3 @@
-##Lab2
-#Submodules in lab:
+# Lab2
+## Submodules in lab:
        geometric_lib
```

4. Создание веток release и hotfix

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git flow release start Release_Maksim
Switched to a new branch 'release/Release_Maksim'

Summary of actions:
- A new branch 'release/Release_Maksim' was created, based on 'develop'
- You are now on branch 'release/Release_Maksim'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git flow release finish 'Release_Maksim'

maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git flow hotfix start 1
warning: unable to rmdir 'geometric_lib': Directory not empty
Switched to a new branch 'hotfix/1'

Summary of actions:
- A new branch 'hotfix/1' was created, based on 'master'
- You are now on branch 'hotfix/1'

Follow-up actions:
- Start committing your hot fixes
- Bump the version number now!
- When done, run:

     git flow hotfix finish '1'
```

5. Клонирование сабмодуля geometric_lib, доабвление gitignore файла, второй тег

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git submodule add https://github.com/ejatohvee/geometric_lib
Cloning into '/Users/maksimignatov/Documents/Gitlab2/geometric_lib'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 1), reused 1 (delta 0), pack-reused 5
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (1/1), done.
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git add .gitignore
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git add .gitmodules
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git commit -m "Submodule geometric_lib was added, gitignore was added"
[feature/Feature_Maksim 93e7274] Submodule geometric_lib was added, gitignore was added
 3 files changed, 5 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .gitmodules
 create mode 160000 geometric_lib
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git tag -a v1.1 -m "Submodule geomrtric_lib"
```

6. Добавление в проект расширения LFS, его настройка на считывание jpg файлов, добавление файлов.

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git lfs install
Updated Git hooks.
Git LFS initialized.
```

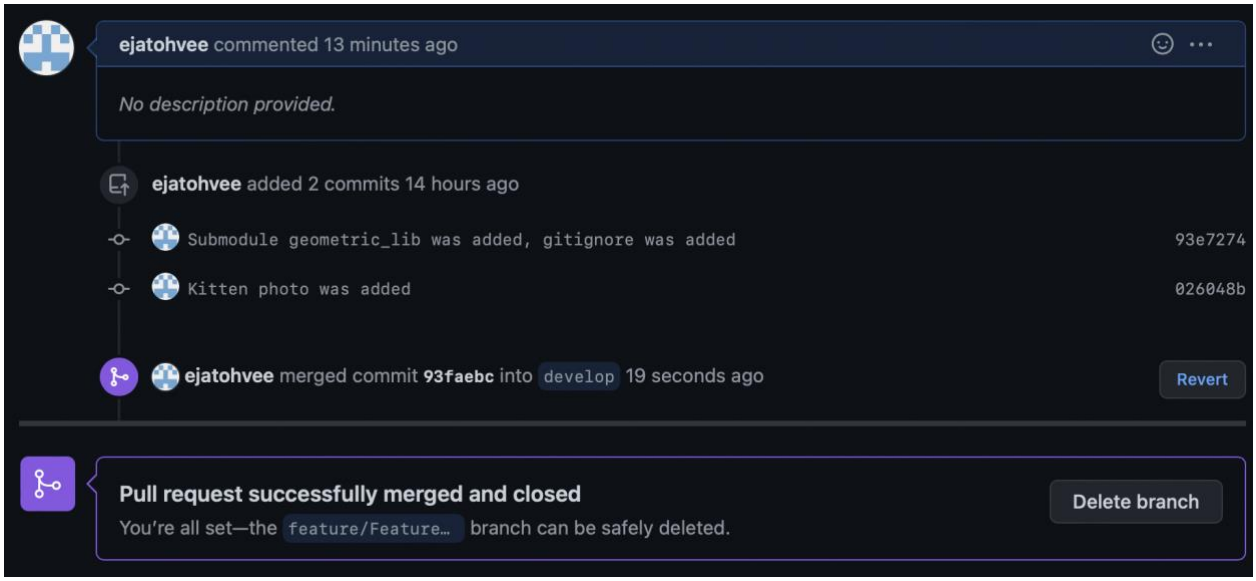
```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git lfs track "*.jpg"
Tracking "*.jpg"
```

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git add .gitattributes
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git add kitten_runs.jpg
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git status
On branch feature/Feature_Maksim
Your branch is up to date with 'origin/feature/Feature_Maksim'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitattributes
        new file:   kitten_runs.jpg

maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git commit -m "Kitten photo was added"
[feature/Feature_Maksim 026048b] Kitten photo was added
 2 files changed, 4 insertions(+)
 create mode 100644 .gitattributes
 create mode 100644 kitten_runs.jpg
```

7. Pull request и merge через Github



8. Клонирование репозитория напарника и его инициализация

```
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % git clone https://github.com/ejatohvee/lab2git.git
Cloning into 'lab2git'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 14 (delta 1), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (1/1), done.
maksimignatov@MacBook-Air-Maksim-3 Gitlab2 % cd lab2git
maksimignatov@MacBook-Air-Maksim-3 lab2git % git log
commit bae82cb9e860819f9f7bea9ba6159ab191e33448 (HEAD -> main, origin/main, origin/HEAD)
Merge: 657c6a9 e56a59e
Author: palmtuft <113020372+palmtuft@users.noreply.github.com>
Date:   Tue Oct 4 12:15:12 2022 +0300

    Merge pull request #1 from palmtuft/feature/Feature_Dusha

    Feature/feature dusha

commit e56a59e22e6c112552a5fef8b08fa93d03d705fd
Author: Dusha <you@example.com>
Date:   Tue Oct 4 12:00:58 2022 +0300

    kitten pounce was added, .gitattributes was added

commit 43bf1ad768de9f0ad37fb2507c02e0596acac1d4
Author: Dusha <you@example.com>
Date:   Tue Oct 4 11:52:18 2022 +0300

    Submodule geometric_lim added, gitmodules added

commit 9636320dcc3e989bc30903d8574fbe10debbfb0f
Author: Dusha <you@example.com>
Date:   Tue Oct 4 00:35:05 2022 +0300

    README.md was modified

commit 657c6a95cb977edf05e08bea5328a3b2972c322e
Author: palmtuft <113020372+palmtuft@users.noreply.github.com>
Date:   Mon Oct 3 20:53:49 2022 +0300
```

9. Создание нового текстового файла и его добавление в репозиторий

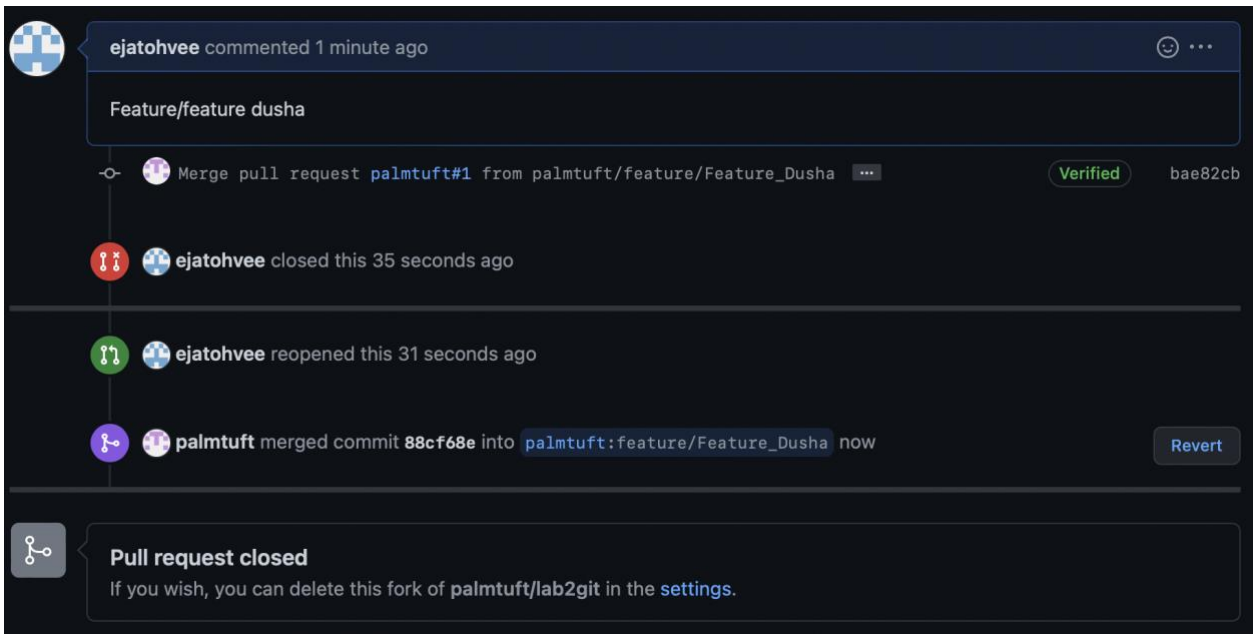
```
maksimignatov@MacBook-Air-Maksim-3 lab2git % touch new_text_file.txt
maksimignatov@MacBook-Air-Maksim-3 lab2git % nano new_text_file.txt
```

```
maksimignatov@MacBook-Air-Maksim-3 lab2git % git add new_text_file.txt
maksimignatov@MacBook-Air-Maksim-3 lab2git % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   new_text_file.txt

maksimignatov@MacBook-Air-Maksim-3 lab2git % git commit -m "New txt file (new_text_file.txt) was added"
[main 0ae0222] New txt file (new_text_file.txt) was added
 1 file changed, 1 insertion(+)
 create mode 100644 new_text_file.txt
```

10. Pull request и merge через Github




# Словарь Git

## Создание Git репозитория

1. cd /Users/user/my_project – переходим в проект, находящийся в указанной папке (если он есть).
2. git init - создание в текущем каталоге подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория (проект еще не под версионным контролем).
3. git clone - гит получает копию практически всех файлов, находящихся на сервере. При выполнении git clone с сервера забирается (pulled) каждая версия каждого файла из истории проекта.

## Запись изменений в репозиторий

1. git status (-s - более сокращенный вариант) - основной инструмент, используемый для определения, какие файлы в каком состоянии находятся.
2. git add - Это многофункциональная команда, она используется для добавления под версионный контроль новых файлов, для индексации изменений, а также для других целей, например для указания файлов с исправленным конфликтом слияния. Если вы изменили файл после выполнения git add, вам придётся снова выполнить git add, чтобы проиндексировать последнюю версию файла.
3. <> .gitignore - создание файла с перечислением шаблонов, которые стоит игнорировать.

$ cat .gitignore *.[oa] - игнорирование файлов, оканчивающихся на .o/.a.

1. git diff - показывает добавленные и удаленные строки в файлах. Показывает непроиндексированные изменения.
    1. git diff - просмотр того что изменено но пока не проиндексировано.
    2. git diff --staged - посмотреть что проиндексировано и войдет в коммит.
    3. git diff --cached - для просмотра проиндексированных изменений (после индексации файла).

2. git commit - фиксация изменений.
    1. git commit -m <>
    2. git commit -a - для быстрого добавления файла в коммит без git add
3. git rm - удаление файла из отслеживаемых (удаляет при этом и из рабочего каталога)
    1. git rm --cached <> - удалить файл из индекса, но оставить в каталоге (остается на жестком диске но изменения не отслеживаются)
4. git mv <было> <стало> - переименование файла и его индексация для следующего коммита (1 команда вместо mv, git rm, git add)

**Просмотр истории коммитов и операции отмены**

1. git log - просмотр истории коммитов
    1. git log -p (--patch) (-2 - последние 2)- показывает разницу (выводит патч), внесенную в каждый коммит
    2. git log --stat - печатает под каждым из коммитов список и количество измененных файлов, а также сколько строк в каждом из файлов было добавлено и удалено
    3. git log pretty=format” ” - смена формата вывода:
        1. %H - хэш коммита, %T - хэш дерева, %P - хэш родителей, %an - имя автора, %s - содержание.

           Автор — это человек, изначально сделавший работу, а коммитер — это человек, который последним применил эту работу.

    4. git log --graph - ASCII граф (git log --pretty=format:"%h %s" --graph)
    5. git log since / untill - вывод, ограниченный по времени
    6. git log -S <file> - фильтр, показывающий только коммиты, в которых был изменен данный файл
    7. git log -- <path/to/file> - показывает только коммиты, в которых было изменение этого файла
2. git commit --amend - дополнение коммита (после добавления с помощью git add)
3. git reset HEAD <file> / git restore --staged <file> - исключение файла из индекса (убрать его из changes to be commited)
4. git checkout -- <file>- удаляет незакоммиченные изменения в самом файле (!опасно потерей данных)
5. git restore <file> - откат файла до предыдущей закоммиченной версии (!опасно потерей данных)

**Работа с удаленными репозиториями**

1. git remote (-v для чтения адресов)- список удаленных репозиториев
2. git remote add <shortname> <url> - добавление удаленного репозитория. В дальнейшем, чтобы получить изменения, находящиеся у кого-то другого, можно воспользоваться кратким именем (git fetch <shortname>)
3. git fetch - получение данных из удаленного репозитория. Изначально просто добавляет файлы в проект (слияние проводится вручную)
4. git fetch origin - извлекает все наработки, отправленные на сервер после того, как вы его клонировали

5. git pull - получение изменений из удаленной ветки и их автоматическое слияние с текущей (в отличие от git clone)
6. git push <remote-name> (origin – репозиторий, из которого изначально клонирован проект) <branch-name> - отправка наработок в удаленный репозиторий
7. git remote show <remote> - информация об удаленном репозитории (URL, отслеживаемые ветки)
8. git remote rename <was> <now> - переименование удаленного репозитория
9. git remote rm <name> - удаление удаленного репозитория

**Работа с тегами**

1. git tag - просмотр тегов
2. git tag -a (-m “ ” для сообщения)- аннотированный тег (хранится в базе данных как полноценный объект. Имеет контрольную сумму, имя автора, email и дату создания, комментарий)
3. git show - просмотр данных тега вместе с коммитом
4. git tag <name> - легковесный тег, с данными о контрольной сумме коммита
5. git tag -a v1.2 <часть контрольной суммы коммита> (первые 7 элементов) - тэг на уже существующий коммит
6. git push origin <tagname> (--tags - всех сразу) - отправка тега на удаленный сервер
7. git tag -d <tagname> - удаление тега в локальном репозитории (не удаляется из удаленного репозитория)
8. git push origin --delete <tagname> - удаление тега с удаленного репозитория
9. git checkout <tag> - переход к версиям файлов, которые были при данном теге (автоматически переход в состояние detached HEAD. Если в состоянии «detached HEAD» внести изменения и сделать коммит, то тег не изменится, при этом новый коммит не будет относиться ни к какой из веток, а доступ к нему можно будет получить только по его хешу)

**Ветвление в Git**

1. git branch - выводит список веток
2. git branch -v - вывод веток с последними коммитами на них
3. git branch --merged (no -merged - не слитые) - показывает слитые в текущую ветку ветки
4. git branch --move - переименование ветки

Ветки - долгоживущие и тематические.

1. git merge - слияние веток

**Подмодули Git**

Для работы с двумя проектами, при этом один находится в другом. Подмодули позволяют вам сохранить один Git-репозиторий, как подкаталог другого Git-репозитория. Это даёт вам возможность клонировать в ваш проект другой репозиторий, но коммиты при этом хранить отдельно.

git submodule add <URL> - добавление подмодуля в репозиторий

Гит распознает подмодуль как комит, пока в него не будет совершен переход.

git push origin master - сохрание изменений в подмодуле

git submodule init - инициализация файлов подмодуля из клонированного репозитория(git clone <URL of project>)

git submodule update - получение всех данных из клонированного проекта и извлечение соответствущего коммита из основного проекта

git fetch → git merge <> / git submodule update --remote - проверка изменений в подмодуле

**LFS Git**

git lfs install - установка

git lfs track "*.psd" - указываем, какие файлы отслеживать (какой формат)

git add .gitattributes - проверка, отслеживаются ли .gitattributes - настройки для отдельных файлов/путей.

**Git Flow**

git flow init – инициаллизация гитфлоу в пректе

git flow feature start <name> - начало новой ветки feature, основанной на ветке develop

git flow feature finish – слияние ветки с develop, ее удаление и переключение на ветку develop

git flow feature publish <name>– публикация ветки на удаленный сервер

git flow feature pull origin <name> - получение опубликованной другим пользователем фичи

git flow feature track – отслеживание фичи

git flow release start <name> [№of commit if you need it] – создание ветки для релиза

git flow release publish <name> - публикация ветки на удаленный сервер

git flow hotfix start <version> [№of commit if you need it] – создание ветки для исправлений

git flow hotfix finish – завершение исправления