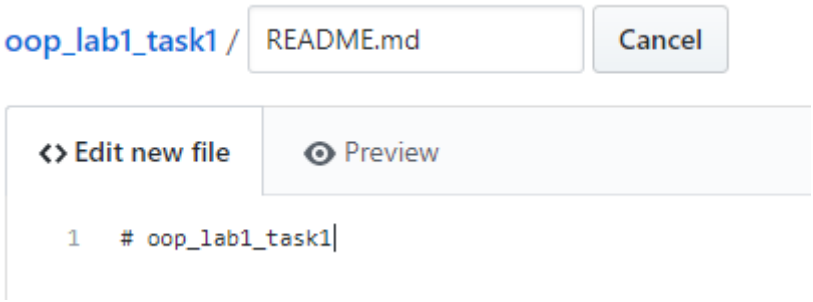
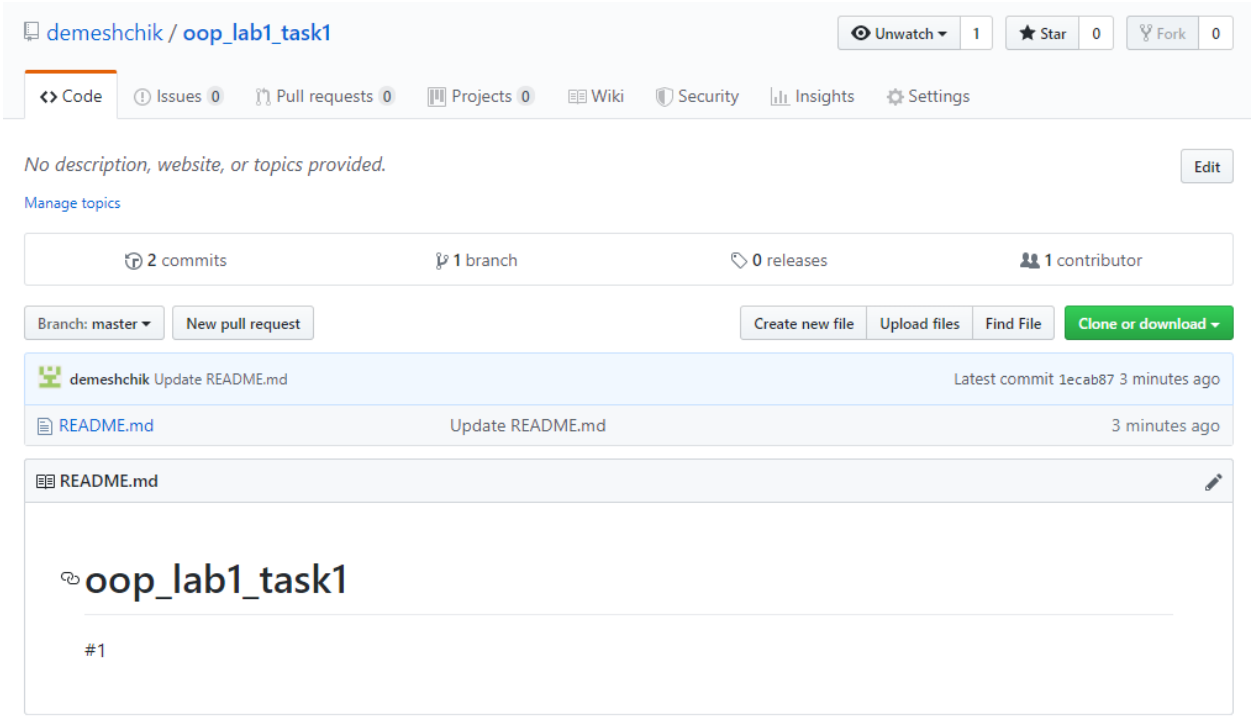
#1. Системы контроля версий

Базовый уровень. Web-интерфейс.

1. Создайте аккаунт на https://github.com
2. Изучите настройки профиля по адресу https://github.com/settings/
3. Создайте первый репозиторий с именем “oop_lab1_task1”
4. Добавьте в репозиторий Readme файл используя web-интерфейс.
5. Вставьте в него свое ФИО. Отформатируйте текст используя Markdown.
   Подсказка:
   - используйте переключатель вверху, чтобы в реальном времени отслеживать как выглядит файл с вашими изменениями.



   - cheatsheet по Markdown - https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
6. Сохраните файл.
7. Изучите обновленный репозиторий.



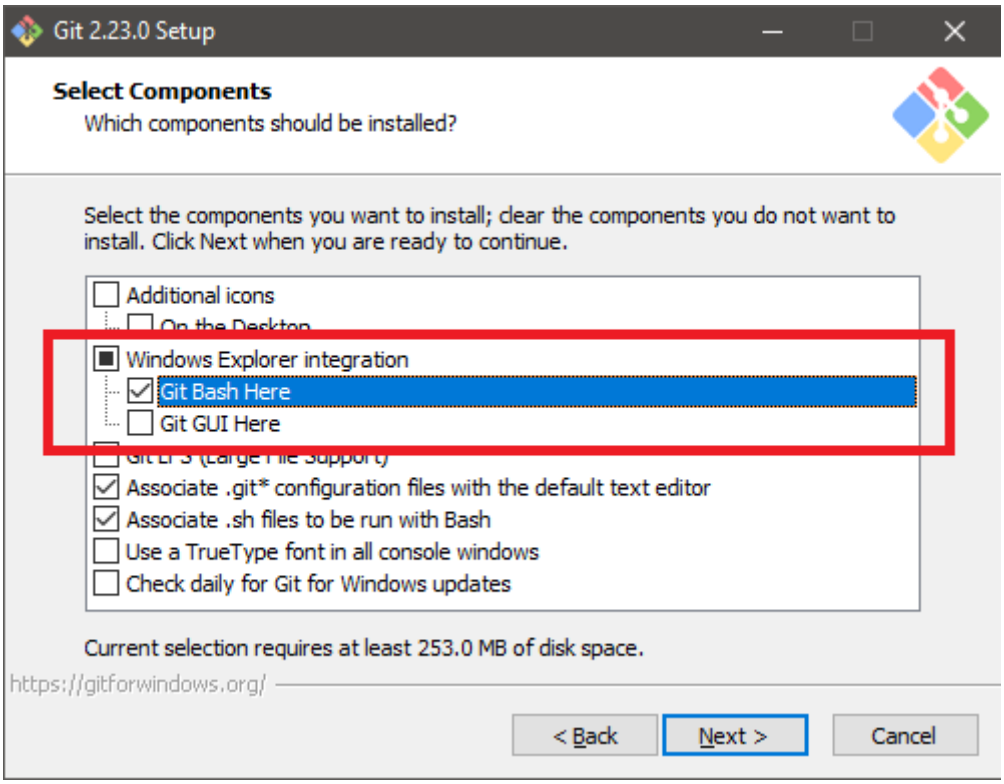
8. Ознакомьтесь с понятиями: коммит, ветка, пул-реквест, хэш коммита, форк, дифф коммита (diff), статус изменений. Эти знания пригодятся при выполнении следующих заданий.

Полезная литература:

a. https://githowto.com/ru
b. https://git-scm.com/book/ru/v2

Базовый уровень. CLI

1. Установите Git для вашей ОС: https://git-scm.com/downloads
2. Во время установки:
   a. Желательно включить Git Bash Here интеграцию с Проводником (Windows). Это облегчит доступ к консоли Гита:



   b. Редактор по умолчанию можно оставить Vim. Он непривычный для windows-пользователей, но он не понадобится во время работы.
   c. Настоятельно НЕ РЕКОМЕНДУЕТСЯ менять настройки для среды PATH. Git по умолчанию предлагает наиболее подходящие.
   d. Рекомендуется оставить дефолтный MinTTY эмулятор.
3. Изучите команды и их предназначения:
   a. git add
   b. git status
   c. git commit

    d. git clone
    e. git pull / git push
    f. git fetch
    g. git init
    h. git config
4. Создайте в своем аккаунте новый пустой репозиторий с именем “oop_lab1_task2”.
5. Запустите Git Bash на компьютере.
6. Создайте новый локальный репозиторий.
7. Обновите конфиг репозитория, установив свою почту и имя.
8. Создайте файл Readme.md с текущей датой и ФИО как содержимым.
9. Создайте коммит с именем “Initial commit. Updated from local repo”, включив в него созданный Readme.md файл.
10. Отправьте изменения в удаленный репозиторий.
    Подсказка:
    - при создании пустого репозитория на сайте github, портал предлагает несколько вариантов инициализации пустого репозитория, один из которых – отправка локального репозитория в удаленный.
11. Измените файл Readme.md через web-интерфейс, добавив текущее время. Сохраните файл.
12. «Заберите» изменения из удаленного репозитория в ваш локальный.
13. Выполните команду git log, проанализируйте ее вывод.
14. Склонируйте репозиторий с именем “oop_lab1_task1”. Измените файл Readme.md, добавив к нему текущую дату.
15. Создайте коммит с именем “Updated Readme.md from cloned repo” и отправьте его в удаленный репозиторий.

Cheatsheet по cli-командам git: https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf

Средний уровень.

1. Изучите предназначение. gitignore файла.
2. Ознакомьтесь с различными шаблонами. gitignore:
    a. https://github.com/github/gitignore
    b. https://www.gitignore.io/
3. Создайте в локальном репозитории “oop_lab1_task2” любой файл и папку.

4. Создайте .gitignore файл.
   Подсказка:
   - чтобы создать в Windows файл с пустым именем, можно добавить точку в конце расширения: ".gitignore.". Windows автоматически удалит последнюю точку, таким образом .gitignore – расширение файла с пустым именем.
5. Обновите созданный. gitignore, добавив в него созданные файл и папку.
6. Зафиксируйте изменения и отправьте их в удаленный репозиторий. Убедитесь, что в удаленном репозитории нет файла и папки, добавленных в. gitignore.

Средний уровень. Ветки

1. Изучите команды и их предназначение:
   a. git branch
   b. git merge
2. Создайте в репозитории "oop_lab1_task2" новую ветку "dev".
3. Создайте в ней любой файл и зафиксируйте изменения в удаленный репозиторий.
4. Смержите dev ветку в master (ветка, создаваемая по умолчанию).
5. Зафиксируйте изменения и отправьте их в удаленный репозиторий.

Средний уровень. Совместная работа

1. Создайте форк репозитория коллеги.
2. Внесите изменения в свой форк (например, добавив какой-нибудь файл).
3. Создайте пулл-реквест в репозиторий коллеги, предложив свои изменения.
4. После мержа ваших изменений в исходный репозиторий его владельцем, заберите в свой форк последние изменения.

Понятный туториал по совместной работе в git - https://www.digitalocean.com/community/tutorials/how-to-create-a-pull-request-on-github

Продвинутый уровень (по желанию):

Ознакомьтесь с командами:

1. git stash

2. git reset
3. git revert
4. git rebase
5. git checkout

Вопросы

1. Что такое git?
2. В чем отличие git от других систем контроля версий?
3. Назовите три возможных состояния файлов в git. В чем их различия?
4. Что такое индекс? Для чего он используется?
5. Какая команда служит для добавления файлов файлов в следующий коммит?
6. Для чего используется команда merge? В чем отличие merge от rebase?
7. Для чего используется команда fetch? В чем отличие команды fetch от pull?
8. Какая команда служит для фиксации изменений?
9. Какая команда позволяет просмотреть информацию о текущем индексе репозитория?
10. Для чего служит команда stash?
11. Какими способами можно создать локальный репозиторий?
12. Для чего нужен .gitignore файл?
13. Для чего используются ветки в git? Как создать/удалить ветку?
14. Каким образом перенести изменения с одной ветки в другую?
15. Какая команда позволяет отобразить удаленные репозитории, связанные с текущим локальным?
16. Каким образом можно получить изменения из удаленного репозитория в локальный?
17. Что такое pull request?
18. Какая команда позволяет отобразить историю репозитория?