

№ 0 Системы контроля версий

Задание

1) **Заведите учетную запись** на сайте <https://github.com/> (если у вас ее нет).

Регистрация на сайте заключается в выборе имени пользователя и пароля. Заполните информацию о себе на странице вашего профиля.

Изучите настройки профиля по адресу <https://github.com/settings>

Создайте репозиторий для вашего первого проекта (можно загрузить проект по ОАП). Для этого нужно перейти по <https://github.com/new>, выбрать название репозитория (произвольным образом), описание и режим доступа "Public".

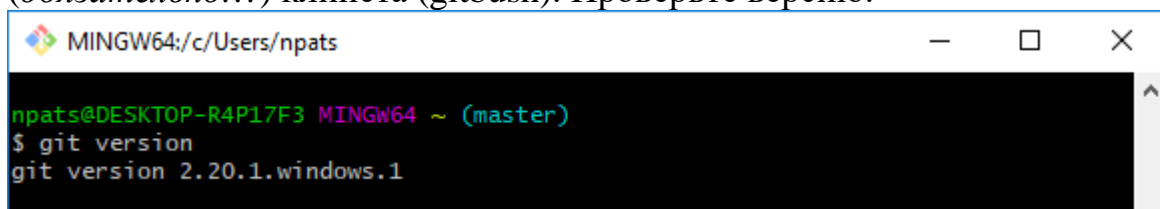
Воспользуйтесь разделом "...or create a new repository on the command line" из подсказки, чтобы инициализировать репозиторий и создать в нем файл README.md

Модифицируйте файл README.md так, чтобы он содержал ваше имя, номер группы и специальность.

Ссылку на получившийся репозиторий нужно прислать преподавателю. Все дальнейшие лабораторные выкладывайте в свой репозиторий для проверки.

2) Базовый уровень. Работа с клиентом GitBush

Скачайте <http://git-scm.com/download/> и установите консольного (обязательно!!!) клиента (gitbush). Проверьте версию.



```
MINGW64: c:/Users/npats
npats@DESKTOP-R4P17F3 MINGW64 ~ (master)
$ git version
git version 2.20.1.windows.1
```

Выполните конфигурацию (git config) (настройте имя пользователя и т.п.).

```
$ git config --global user.name "XXX"
$ git config --global user.email XXX@XXX.com
```

Проверить выбранные настройки позволяет команда `git config --list`, выводящая список всех обнаруженных в текущий момент параметров:

Инициализируйте репозиторий.

Есть два подхода к созданию Git-проекта. Можно взять существующий проект или папку и импортировать в Git. А можно клонировать уже существующий репозиторий с другого сервера.

Научитесь клонировать репозитории (clone). Например, созданный ранее на <https://github.com>.

Научитесь получать информацию о статусе (status) приндексированных и не индексированных файлов. Для этого добавьте в репозиторий новый файл и посмотрите его статус.

Внесите изменения в репозиторий и зафиксируйте изменения.

Настройте gitignore файл для вашего с# репозитория. Перейдите по ссылке <https://github.com/github/gitignore> и изучите структуру нужного файла.

Выполните команды управления и фиксации изменений (add , commit с добавлением сообщения).

Изучите команды удаления и перемещения файлов.

Изучите команду git log.

Команда git log выводит в обратном хронологическом порядке список сохраненных в данный репозиторий версий. То есть первыми показываются самые свежие коммиты. Как видите, рядом с каждым коммитом указывается его контрольная сумма SHA-1, имя и электронная почта автора, дата создания и сообщение о фиксации.

Сделайте отмену внесенных в файл изменений.

git checkout -- [file] — опасная команда. Любые изменения соответствующего файла пропадают — вы просто копируете поверх него другой файл. Ни в коем случае не используйте эту команду, если вы не убеждены, что файл вам не нужен.

Все, что зафиксировано коммитом в Git, почти всегда можно восстановить. Но все, что вы потеряете, не сделав коммит, скорее всего, вам больше не увидеть.

Изучите команды **reset** и **revert**

3) Работа с ветками

Создайте (branch) новую ветку, например, test и переключитесь (checkout) в нее. Добавьте новый файл test.json и выполните commit.

Выполните слияние (merge) ветвей.

По очереди отредактируйте файл test.json в ветках master и test. Выполните commit. Выполните слияние. Разрешите ситуацию конфликта слияний.

4) Работа с удаленными репозиториями

Выполните команды загрузки (push) и выгрузки (pull, fetch) в удаленный репозиторий.

5) Совместная работа

У вас есть репозиторий, в который будут приходить pull request и фиксироваться владельцем репозитория.

Найдите репозиторий коллеги над которым вы хотели бы вести совместную работу. Найдите кнопку *Fork* и нажмите ее. После этого создается копия этого репозитория, но уже в вашем аккаунте.

И теперь этот уже свой репозиторий клонируем себе на локальную машину. Внесите изменения в файл.

Сделайте commit.

Отправьте изменения в свой удаленный облачный репозиторий.

Создаем Pull requests через интерфейс Github. Для этого есть кнопка, которая так и называется. Нажимаем кнопку Create pull request.

После этого переходим на страницу репозитория с которым мы хотим объединиться. Видно кто и какие изменения предлагается внести.

Ваш коллега должен зафиксировать изменения.

Теперь владелец репозитория внесет изменения и нам тоже надо их получить. Для этого сделаем новую ветку и переключимся в нее. Зафиксируем изменения сделанные в совместном проекте у себя

6) Интеграция со средой разработки

Используя инструменты интеграции с git выполните предыдущие задания, но из среды разработки (Visual Studio xxx).

7) Дополнительно (по желанию) изучите команды

git remote

git stash

git rebase

git revert

git tag

Используйте в папке Литература: книги по Git

<https://githowto.com/ru>

<https://git-scm.com/book/ru/v2>

<https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>

Посмотрите видео

<https://elearn.epam.com/courses/course-v1:EPAM+VCG+RU/course/>

Вопросы

1. Что такое система контроля версий, для чего ее используют?
2. Какой принцип хранения файлов использует Git?
3. В чем отличие git от других систем контроля версий?
4. В каких трех основных состояниях файлы могут находиться в Git ?
5. Что такое индексация файла? Какой командой она выполняется?
6. Что такое фиксация файла? Какой командой она выполняется?
7. Продемонстрируйте команду проверки выбранных настроек.
8. Как инициализировать репозиторий в существующей папке?
9. Как указать файлы, за которыми должна следить система?
10. Как выполнить фиксацию изменений?
11. Какой командой определить состояния файлов?
12. Для чего создается файл .gitignore? Поясните его структуру.
13. Как используется команда git diff?
14. Как используется команда git commit?
15. Как используется команда git log? Какие у нее есть параметры?

16. Как используется команда `git commit –amend`?
17. Как отобразить удаленные репозитории?
18. Как извлечь данные из удаленного репозитория?
19. Как отправить данные в удаленный репозиторий?
20. Какая команда позволяет отобразить удаленные репозитории, связанные с текущим локальным?
21. Каким образом можно получить изменения из удаленного репозитория в локальный?
22. Для чего используется команда `fetch`? В чем отличие команды `fetch` от `pull`?
23. Для чего используется команда `merge`? В чем отличие `merge` от `rebase`?
24. Что такое `pull request`?
25. Какая команда позволяет отобразить историю репозитория?
26. Продемонстрируйте создание новых веток в Git? Что такое ветвление? Что такое указатель HEAD?
27. Как используется команда `git checkout`?
28. Как выполнить включение изменений из одной ветки в другую?
29. Какие проблемы могут быть при слиянии и как они разрешаются?
30. Что такое GitLab?