Лекция 2
Работа с GIT

ПМ.02 Разработка, адаптация и внедрение ПО отраслевой направленности

МДК 02.01 Раздел 2 Основы программирования информационного контента на ЯВУ

Тимашева Эльза Ринадовна

Для чего это нужно?

Управлять версиями ПП, место, сохраняются только(!) изменения, сохранение не на своем ПК – совместная работа над проектом, не забивается свой жесткий диск)

Удаленное хранилище версий

2005, Линус Торвальдс.

Pro Git by Scott Chacon and Ben Straub is available to read online for free.
Dead tree versions are available on Amazon.com.

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

Установка git

https://git-scm.com/

распределённые системы контроля версий (РСКВ).
В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) — они полностью копируют репозиторий.
В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы.
Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.
Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён.
Git представляет свои данные как, скажем, **поток снимков**.

У Git есть три основных состояния, в которых могут находиться ваши файлы:
изменён (modified),
индексирован (staged) и
зафиксирован (committed):

К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы.

Индексированный — это изменённый файл в его текущей версии, отмеченный для включения в следующий коммит.

Зафиксированный значит, что файл уже сохранён в вашей локальной базе.

Три основные секции проекта Git:
рабочая копия (working tree),
область индексирования (staging area) и
каталог Git (Git directory).

Рабочая копия является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.

Область индексирования — это файл, обычно находящийся в каталоге Git, в нём содержится информация о том, что попадёт в следующий коммит. Её техническое название на языке Git — «индекс», но фраза «область индексирования» также работает.

Каталог Git — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git и это та часть, которая копируется при **клонировании** репозитория с другого компьютера.

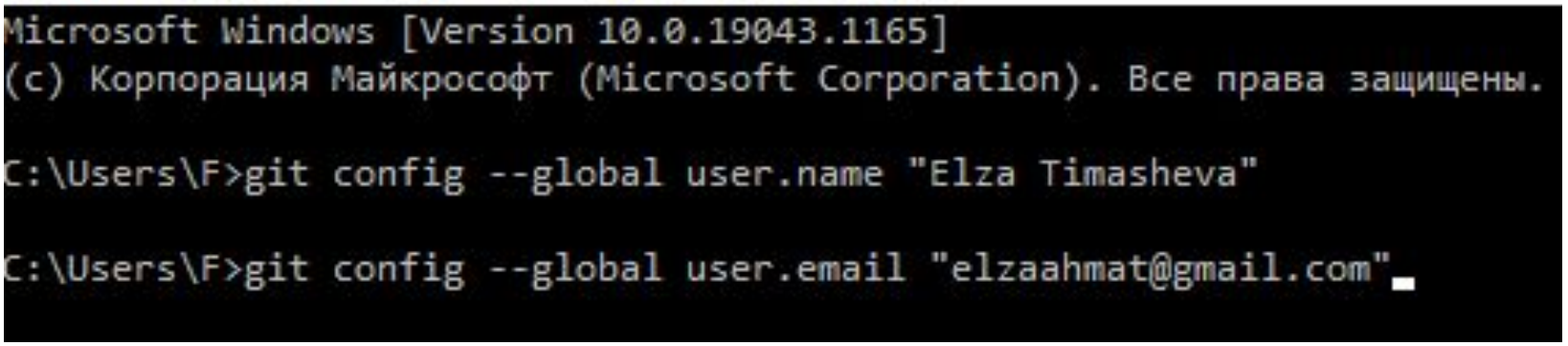
Базовый подход в работе с Git выглядит так:

- Изменяете файлы вашей рабочей копии.
- Выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит, добавляя тем самым снимки **только** этих изменений в индекс.
- Когда вы делаете коммит, используются файлы из индекса как есть, и этот снимок сохраняется в ваш каталог Git.

Если определённая версия файла есть в каталоге Git, эта версия считается **зафиксированной** (committed).

Если файл был изменён и добавлен в индекс, значит, он **индексирован** (staged).

Если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он



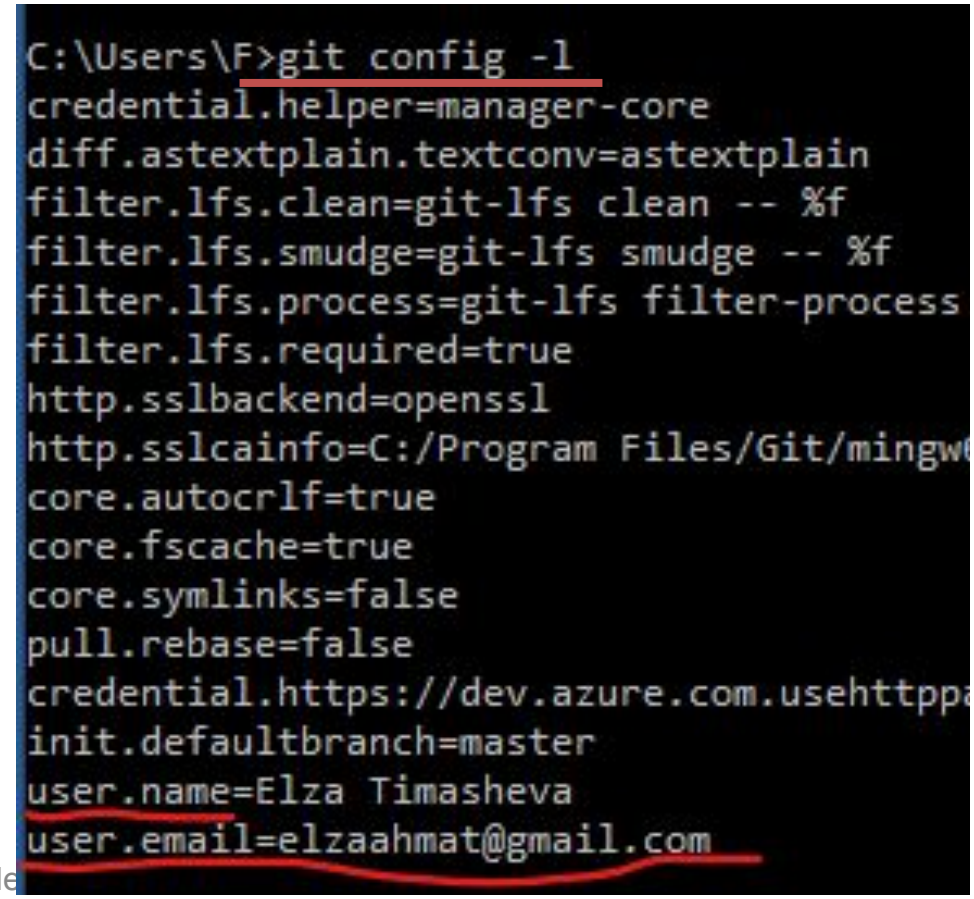
git config --list --show-origin

Выполнить первоначальные настройки конфигурации:

```
git config --global
user.name "your name"

git config --global
user.email "your e-mail"
```

Посмотреть:

`git config –list` *или* `git config -l`



Создание Git-репозитория

Обычно репозиторий Git получают одним из двух способов:

1) взять локальный каталог, который в настоящее время не находится под версионным контролем, и превратить его в репозиторий Git,
2) клонировать существующий репозиторий Git из любого места.

В обоих случаях получаете готовый к работе Git-репозиторий на компьютере.

Создадим репозиторий для работы.

Создавать будем на диске D (ОС Windows).

Последовательность команд	*Значение команды*
`d:`	Поменяли директорию с C на D (необязательно делать)
`md myproject`	Создаем рабочую папку с именем myproject (напр.)
`cd myproject`	Заходим в рабочую папку (ее будем помещать под версионный контроль)
`git init .`	Создали структуру **git-репозитория**
`dir .git`	Посмотрели содержимое папки git (она скрытая)
`git status`	Определение состояния (**статуса**) файлов

`copy con file1.txt`
`Привет`
`^Z`
	Создать файл file1.txt
	Заполнить его строкой текста «Привет»

`git status`	Определение состояния (**статуса**) файлов
`git add *`	Добавили файлы в **индекс** (все, какие в папке были), т.е. проиндексировали файл для **добавления его в следующий коммит**
`git status -s`	Определение состояния (**статуса**) файлов – краткая версия

```
d:\myproject>git init .
Initialized empty Git repository in D:/myproject/.git/
```

```
d:\myproject>dir .git
 Том в устройстве D имеет метку DATA
 Серийный номер тома: A062-1736

 Содержимое папки d:\myproject\.git

14.09.2021  21:12               130 config
14.09.2021  21:12                73 description
14.09.2021  21:12                23 HEAD
14.09.2021  21:12    <DIR>          hooks
14.09.2021  21:12    <DIR>          info
14.09.2021  21:12    <DIR>          objects
14.09.2021  21:12    <DIR>          refs
```

git help <команда> - справка по команде

```
d:\myproject>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

```
D:\myproject>copy con file1.txt
привет
```

```
D:\myproject>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Untracked – неотслеживаемый.

Статус Untracked означает, что Git видит файл, которого не было в предыдущем снимке состояния (коммите);

Git не станет добавлять его в ваши коммиты, пока вы его явно об этом не попросите.

```
D:\myproject>git add *
```

```
D:\myproject>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
```

Теперь файл стал отслеживаемым.

Если вы изменили файл после выполнения git add, вам придётся снова выполнить git add, чтобы проиндексировать последнюю версию файла.

```
D:\myproject>git status -s
A  file1.txt
```

Последовательность команд	*Значение команды*
`git commit -m "My first commit"`	Закоммитили файлы (комментарий обязателен)
`git status`	Определение **статуса** файлов
`echo File2 > file2.txt` `echo Файл3 > file3.txt`	Создаем еще пару текстовых файлов
`git status`	Определение **статуса** файлов
`git add *`	Добавили файлы в **индекс** (все, какие в папке были), т.е. проиндексировали файл для **добавления его в следующий коммит**
`git status`	Определение **статуса** файлов
`Изменили содержимое файла file1.txt`	Можно через проводник (быстрее)
`git status`	

```
D:\myproject>git commit -m "My first commit"
[master (root-commit) 4fbf646] My first commit
 1 file changed, 3 insertions(+)
 create mode 100644 file1.txt
```

коммит вывел немного информации о себе: на какую ветку выполнили коммит (master), какая контрольная сумма SHA-1 у этого коммита (4fbf646), сколько файлов было изменено, а также статистику по добавленным/удалённым строкам в этом коммите.
Запомните, что коммит сохраняет снимок состояния индекса. Всё, что НЕ проиндексировали, так и висит в рабочем каталоге как изменённое; можно сделать ещё один коммит, чтобы добавить эти изменения в репозиторий.

```
D:\myproject>git status
On branch master
nothing to commit, working tree clean
```

```
D:\myproject>echo File2 > file2.txt
```

```
D:\myproject>echo Файл3 > file3.txt
```

```
D:\myproject>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt
        file3.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Каждый раз, когда делаете коммит, вы сохраняете **снимок состояния проекта**, который позже можете восстановить или с которым можно сравнить текущее состояние.

```
D:\myproject>git add *

D:\myproject>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2.txt
        new file:   file3.txt
```

```
D:\myproject>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2.txt
        new file:   file3.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:   file1.txt
```

Последовательность команд	*Значение команды*
`git commit -m "Added file2,file3, modified file1"`	Закоммитили файлы (комментарий обязателен)
`git status`	Определение **статуса** файлов
`git add *`	Добавили файлы в **индекс** (все, какие в папке были), т.е. проиндексировали файл для **добавления его в следующий коммит**
`git commit -m "Modified file1"`	Закоммитили файлы (комментарий обязателен) Теперь и измененный файл закоммититься
	пределение **статуса** файлов

```
D:\myproject>git commit -m "Added file2,file3, modified file1"
[master fa9f02e] Added file2,file3, modified file1
 2 files changed, 2 insertions(+)
 create mode 100644 file2.txt
 create mode 100644 file3.txt

D:\myproject>git status
On branch master
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")
```

```
D:\myproject>git add *

D:\myproject>git commit -m "Modified file1"
[master 1e48377] Modified file1
 1 file changed, 1 insertion(+)

D:\myproject>git status
On branch master
nothing to commit, working tree clean
```

команда git

- **`echo "text" > file.txt`**
  - untracked
  - неотсле-живаемый

- **`git add`**
- **`file.txt`**
  - staged
  - индекси-рованный

- **`git commit`**
- **`–m “comment”`**
  - copy in local repository
  - закоммитили, сделали снимок версии

- **`git push`**
  - copy in Remote Repository
  - запУшили

состояние контента

Search or jump to... / Pull requests Issues Marketplace Explore

ElzaTimasheva / firstproject Public

Забираем клон проекта в свой локал.репозиторий для работы

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

ElzaTimasheva Initial commit

README.md Initial commit

README.md

firstproject

Тренируемся

Clone

HTTPS SSH GitHub CLI New

https://github.com/ElzaTimasheva/firstpro

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

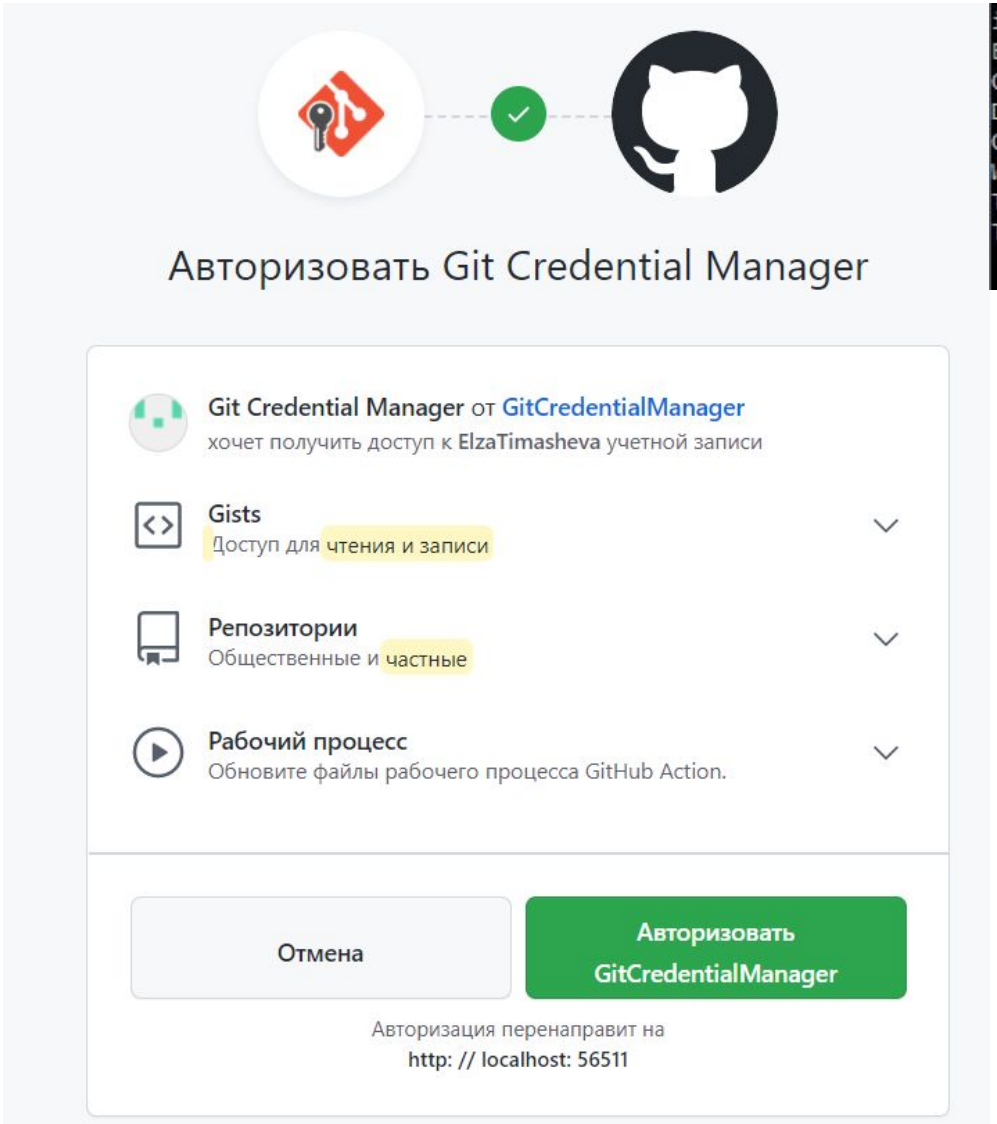
В командной строке прописываем:
git clone https://github.com/ElzaTimasheva/firstproject.git

```
D:\myproject>git clone https://github.com/ElzaTimasheva/firstproject.git
Cloning into 'firstproject'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

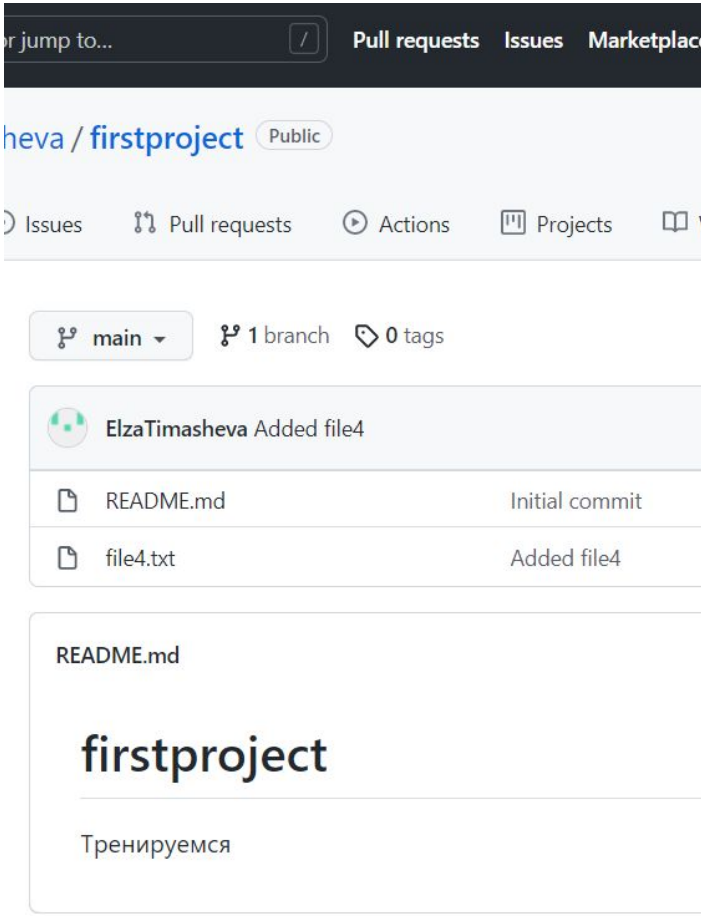
й компьютер › Локальный диск (D:) › myproject ›

Имя	Дата и
.git	15.09.2
firstproject	15.09.2
file1.txt	15.09.2
file2.txt	15.09.2
file3.txt	15.09.2

```
D:\myproject>cd firstproject

D:\myproject\firstproject>echo bla=bla-bla > file4.txt

D:\myproject\firstproject>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file4.txt

nothing added to commit but untracked files present (use "git add" to track)

D:\myproject\firstproject>git add *

D:\myproject\firstproject>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:   file4.txt

D:\myproject\firstproject>git commit -m "Added file4"
[main 183f763] Added file4
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt

D:\myproject\firstproject>git push origin
info: please complete authentication in your browser...
```




```
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 58.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ElzaTimasheva/firstproject.git
   d86d078..183f763  main -> main
```



ДЗ

1. Установить git
2. Пройти по командам в таблицах в лекции
3. Зарегистрироваться на GitHub
4. Создать там пустой проект под именем first
5. Клонировать его в свой локалрепозиторий
6. Закинуть в него несколько файлов разного формата (например, ваши любимые мемы)
7. Запушить содержимое на гитхаб
8. Прислать мне ссылку на проверку

https://showskills.ru/it/developer/frontend-middle/33-osnovy-git-i-github.html#eight
видеоуроки по гиту (на линукс, но понять можно)