**Конспект лекции 1**

Можно выделить следующие преимущества систем управления Git-репозиториями:

● Хранение удалённых репозиториев

● Расширенное управление доступами

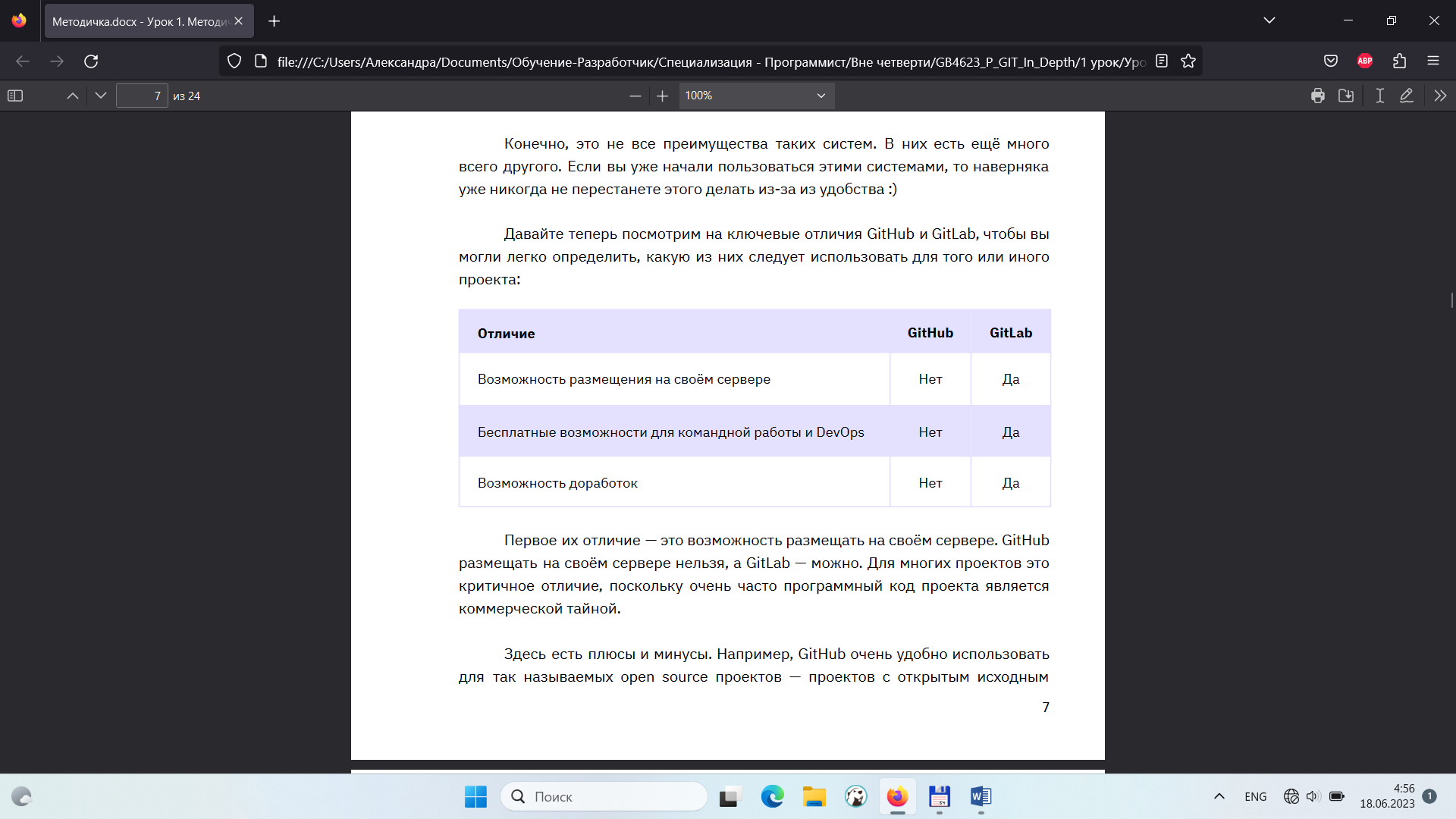
● Наглядное представление репозиториев

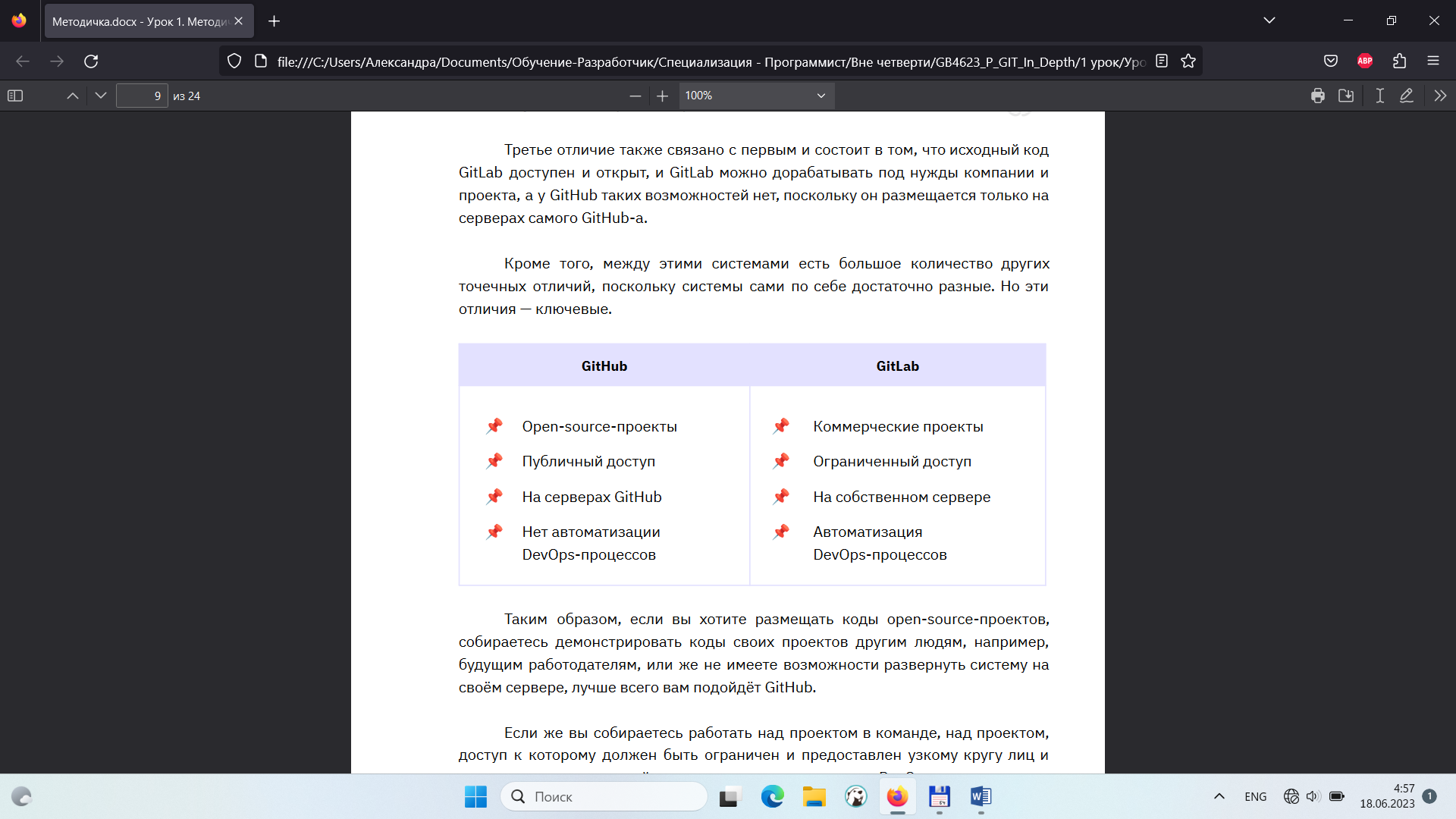
● Управление задачами

● Связь изменений с задачами

● Обсуждение изменений в коде

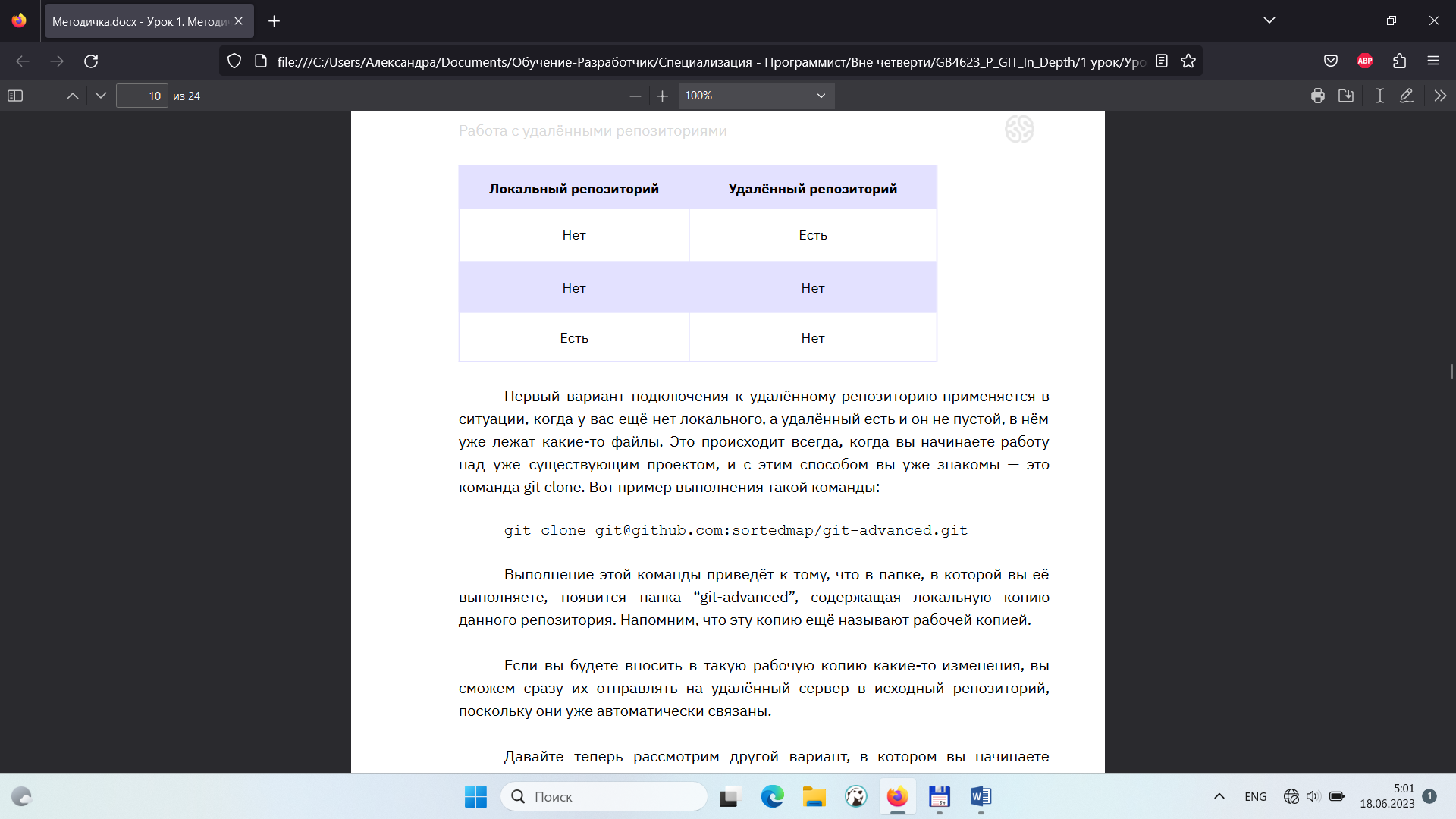
Ключевые отличия GitHub и GitLab:





Подключение к удалённому репозиторию

Несколько способов подключения к удалённым репозиториям в зависимости от того, пустой ли удалённый репозиторий или нет, и создан ли локальный репозиторий.

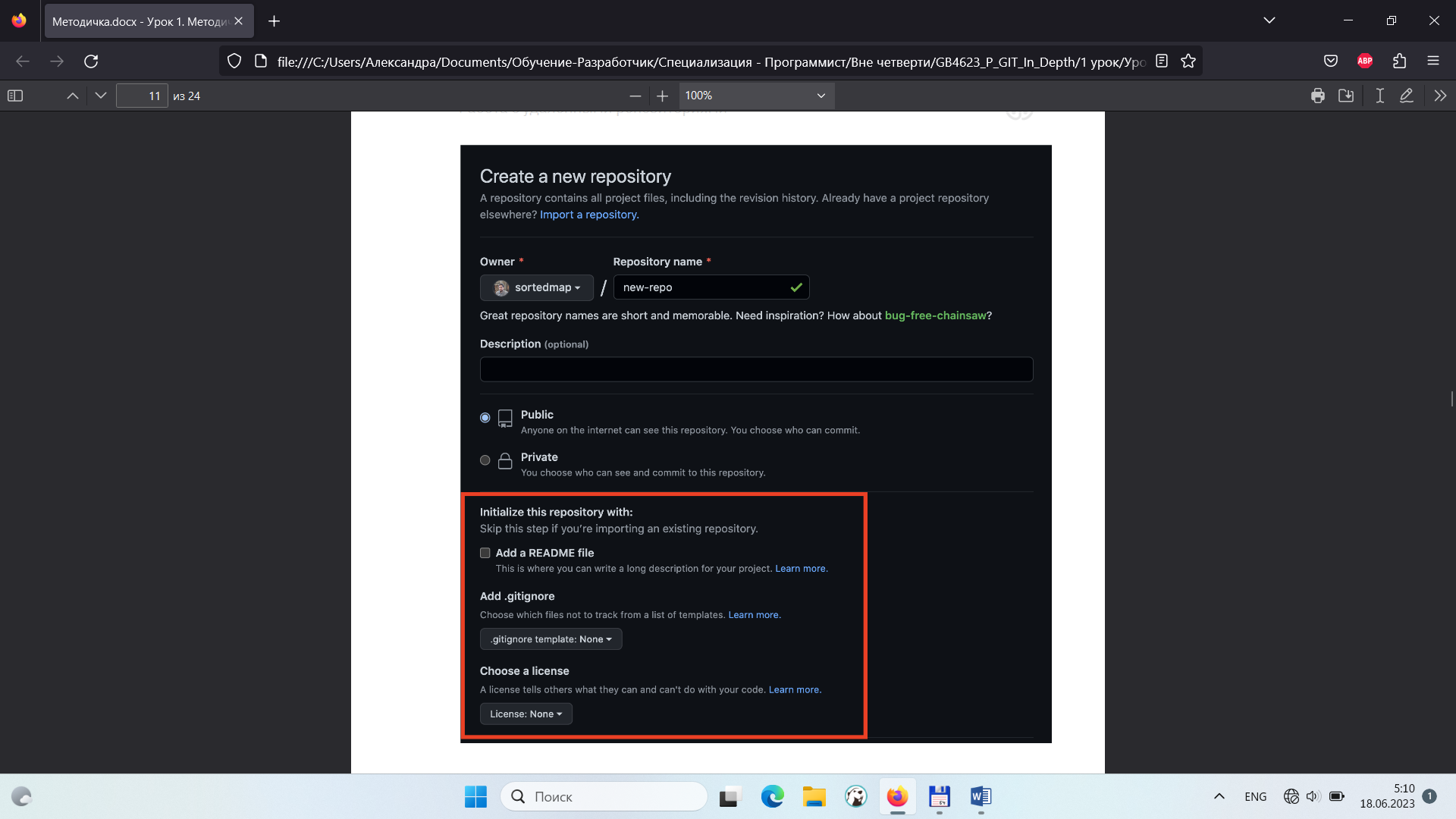


Первый вариант подключения к удалённому репозиторию применяется в ситуации, когда ещё нет локального, а удалённый есть и он не пустой. Это происходит всегда, когда вы начинаете работу над уже существующим проектом.

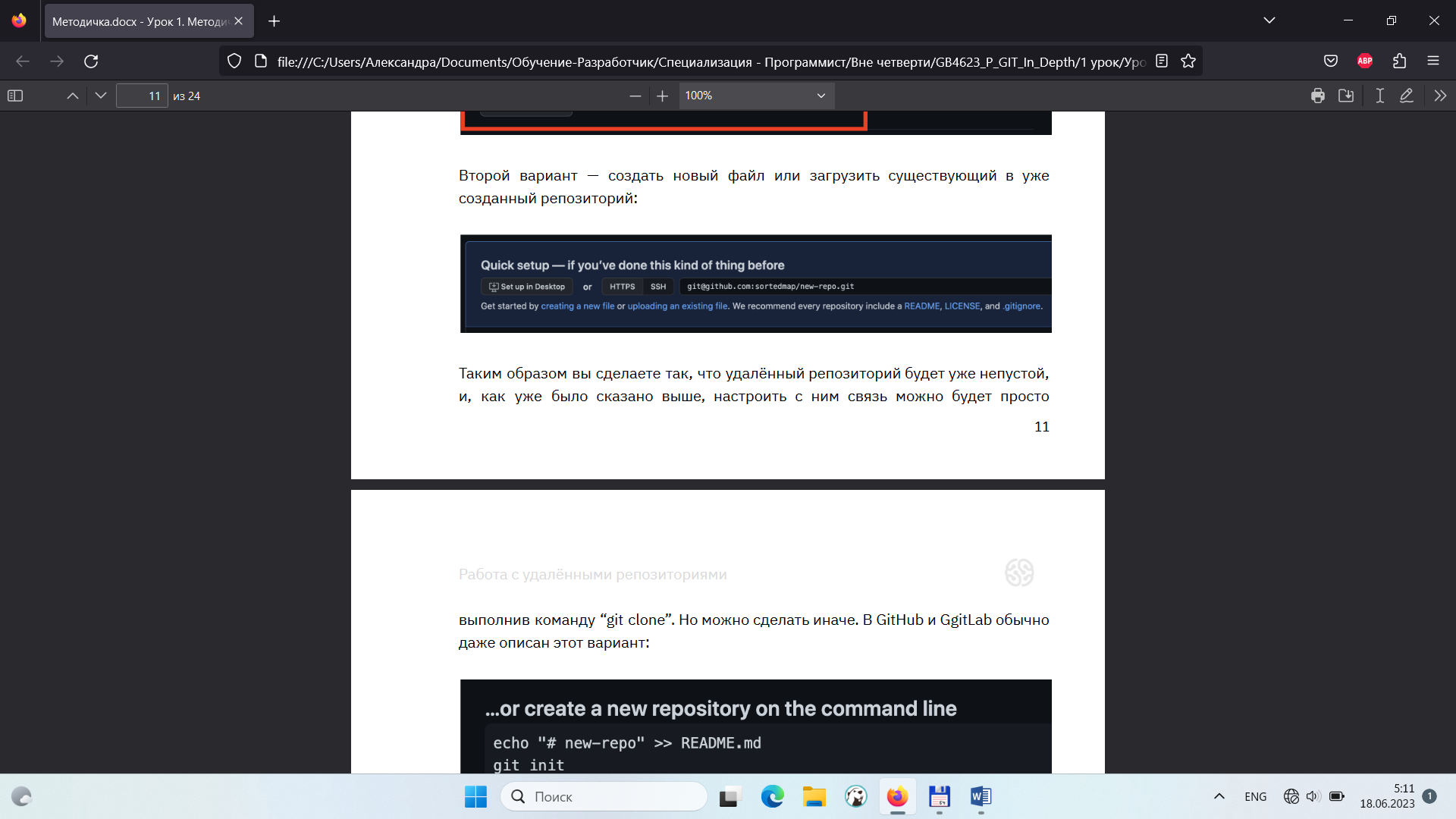
git clone [git@github.com:sortedmap/git-advanced.git](mailto:git@github.com:sortedmap/git-advanced.git)

Второй вариант, в котором вы начинаете работать над новым проектом, и у вас ещё нет ни локального, ни удалённого репозитория. В этом случае нужно сначала создать удалённый репозиторий.

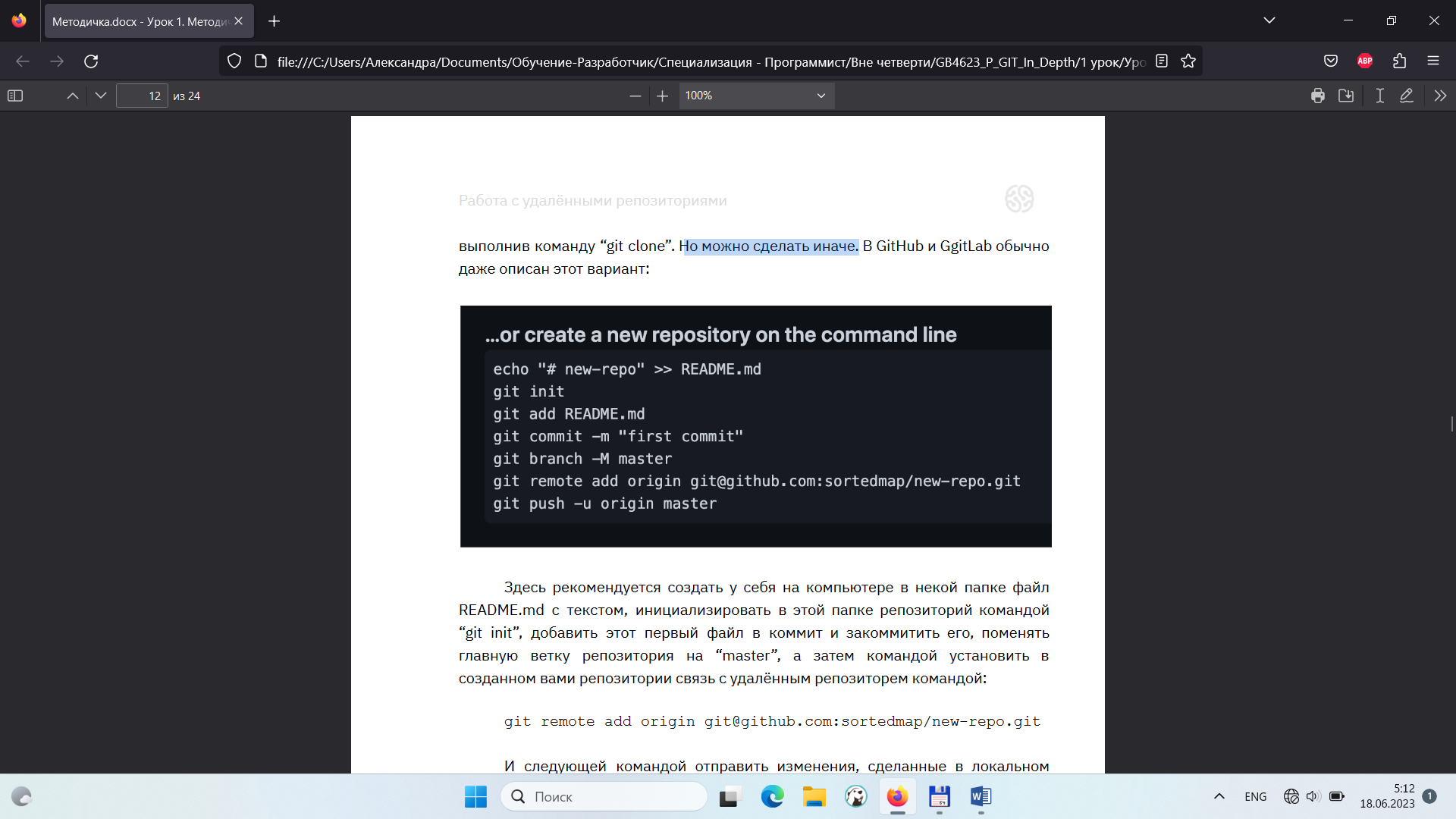
Вы можете сразу добавить в него какие-либо файлы и просто склонировать его к себе на компьютер, как было показано выше, а можете оставить его пустым.



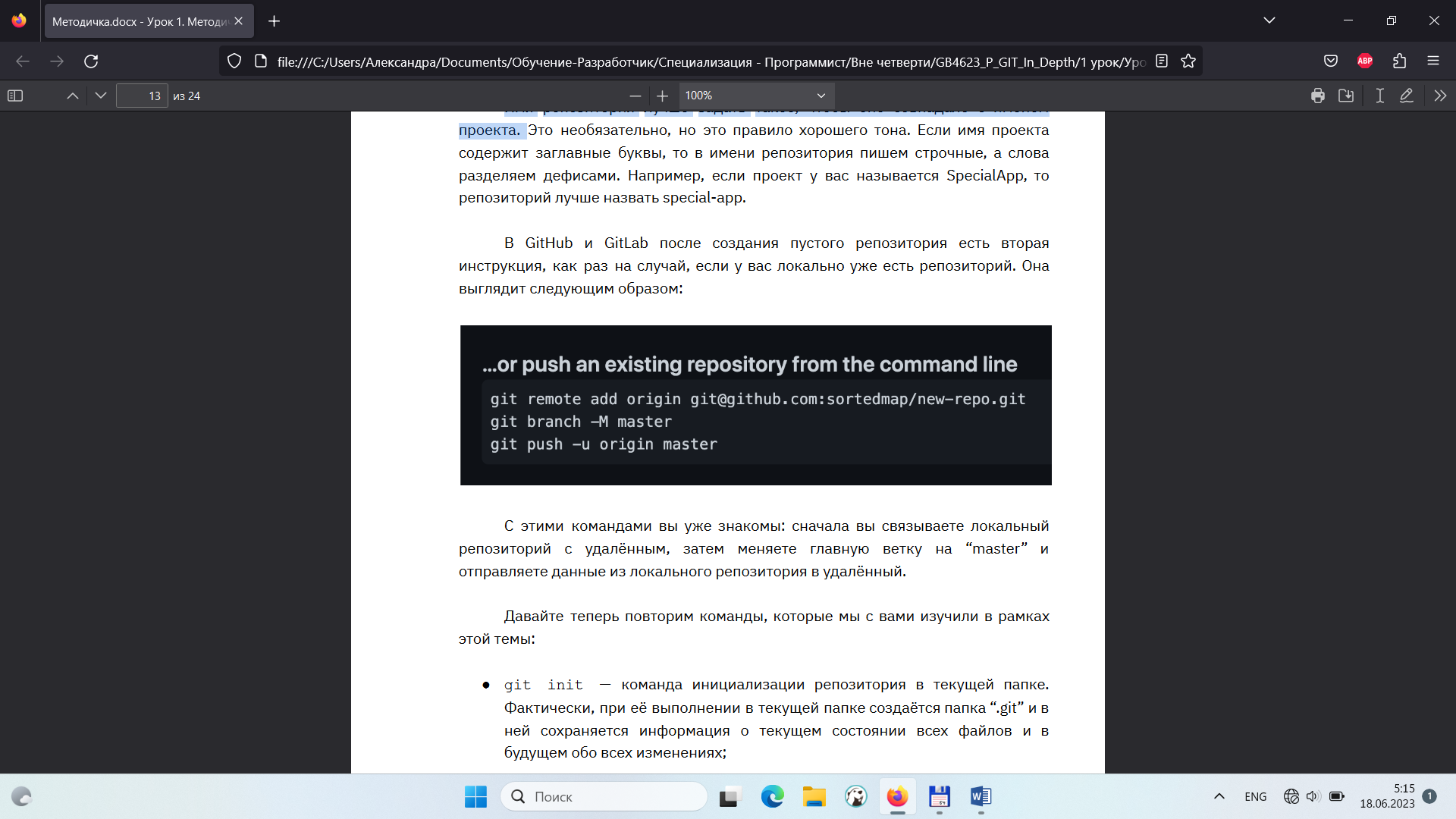
Второй вариант — создать новый файл или загрузить существующий в уже созданный репозиторий:



Но можно сделать иначе.



Третий вариант: у вас уже есть локальный репозиторий с проектом, а удалённого репозитория ещё нет. В этом случае первым действием нужно создать пустой удалённый репозиторий, не добавляя в него никакие файлы. Имя репозитория лучше задать такое, чтобы оно совпадало с именем проекта.

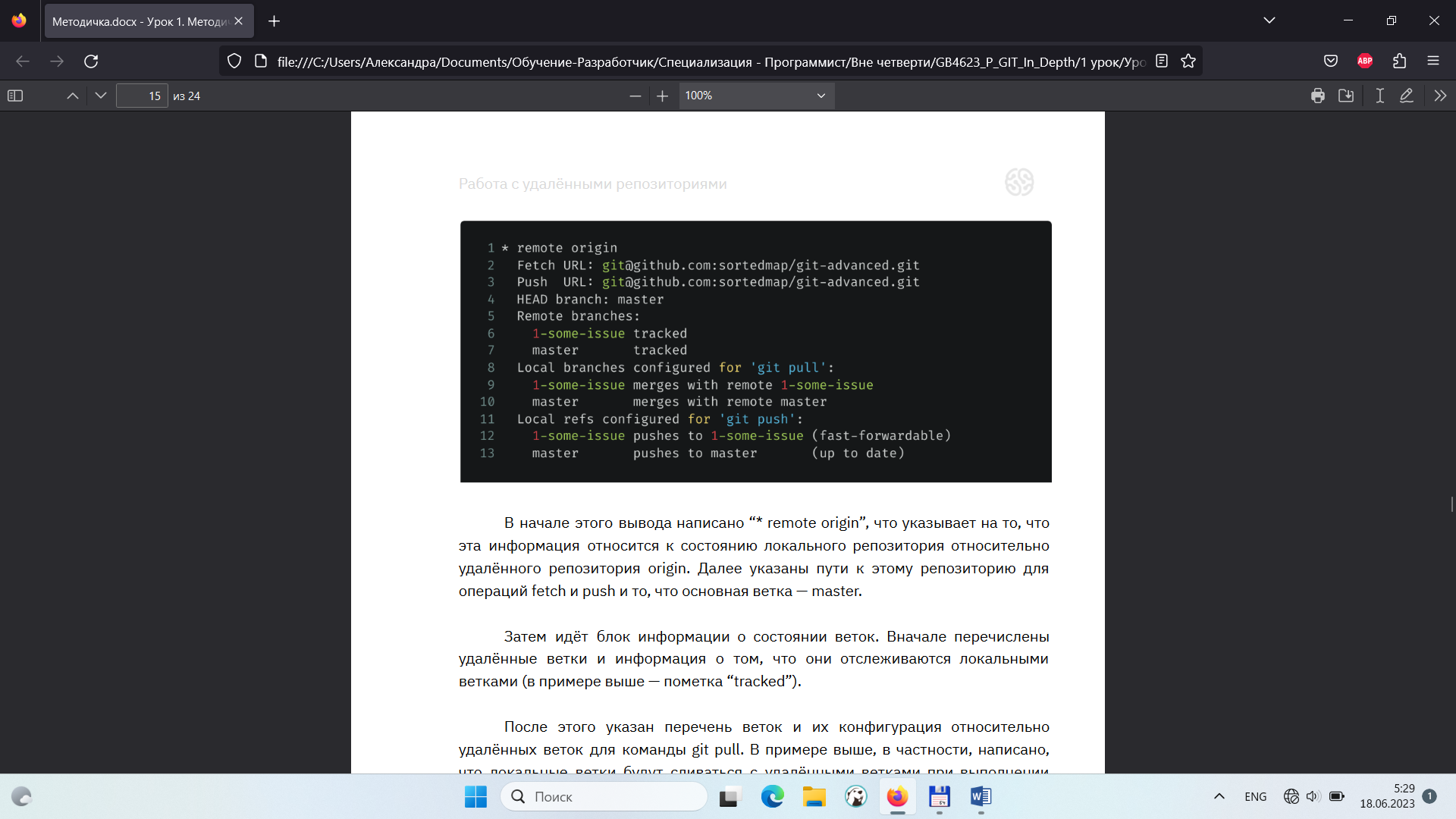


git push -u origin master — команда, которая отправляет изменения из локального репозитория на сервер. Параметр “-u” говорит о том, что необходимо установить связь текущей ветки в локальном репозитории с соответствующей веткой в удалённом таким образом, чтобы вы могли дальше выполнять команду “git push” и некоторые другие команды без дополнительных параметров.

Синхронизация изменений в ветках

git remote show origin

Эта команда позволяет просмотреть, какие локальные ветки связаны с ветками удалённого репозитория и каков их статус. Эта команда выводит информацию в следующем формате:



Вы можете скачать все изменения из ветки удалённого репозитория в локальный, но без обновления самой ветки обновлять не будем. Это делается при помощи команды:

git fetch

После её выполнения вы можете посмотреть, какие изменения были внесены в удалённую ветку, для этого можно воспользоваться командой git log, в параметрах которой указав имена удалённой и локальной веток:

git log origin/1-issue ^1-issue

Такой вариант команды выведет те коммиты, которые отличают удалённую ветку от локальной. Содержимое каждого такого коммита вы можете просмотреть командой:

git diff hash

В данном случае, hash — это хэш коммита. Если вы захотите влить удалённую ветку в свою локальную, вы можете выполнить уже известную вам команду:

git pull

Эту команду также можно выполнять для отдельных веток, указав имя удалённого репозитория и название нужной ветки:

git pull origin 1-issue

Таким образом, команда git pull затягивает изменения из удалённого репозитория и сливает их с изменениями в локальном репозитории, в то время как команда git fetch их просто затягивает, но слияния не производит.

**Практика**

В ветке master в локальном репозитории создадим новый файл CHANGELOG.md с текстом. Создадим коммит.

Далее создадим новую ветку: git checkout -b ‘1-some-issue’. Внесем изменение в файл CHANGELOG.md, добавив новый текст. Добавим коммит.

Отправим в удаленный репозиторий изменения из новой ветки: git push -u origin 1-some-issue.

В результате в удаленном репозитории будет 2 ветки. Файл CHANGELOG.md с изменениями будет только во второй ветке.

Теперь внесем изменения в ветку в удаленном репозитории (как будто там поработал кто-то другой): создадим файл script.js и добавим в него код.

Переходим в ветку master: git checkout master --->>>

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

Проверяем статус: git status --->>>

On branch master

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

Перейдя в ветку 1-some-issue и проверив ее статус, изменений в удаленном репозитории видно не будет: git checkout 1-some-issue --->>>

Switched to branch '1-some-issue'

Your branch is up to date with 'origin/1-some-issue'.

git status --->>>

On branch 1-some-issue

Your branch is up to date with 'origin/1-some-issue'.

nothing to commit, working tree clean

Теперь запросим информацию из удаленного репозитория: git remote show origin --->>>

Local refs configured for 'git push':

1-some-issue pushes to 1-some-issue (local out of date)

master pushes to master (fast-forwardable)

Запросим изменения из удаленного репозитория из ветки 1-some-issue, но без слияния: git fetch

Проверяем статус (изменения еще не внесены): git remote show origin --->>>

Local refs configured for 'git push':

1-some-issue pushes to 1-some-issue (local out of date)

master pushes to master (fast-forwardable)

Проверить какие изменения были сделаны можно с помощью git log origin/1-some-issue ^1-some-issue. Будут видны все коммиты. Чтобы увидеть сами изменения используем команду git diff хэш последнего коммита.

Для затягивания изменения нужно выполнить git pull.

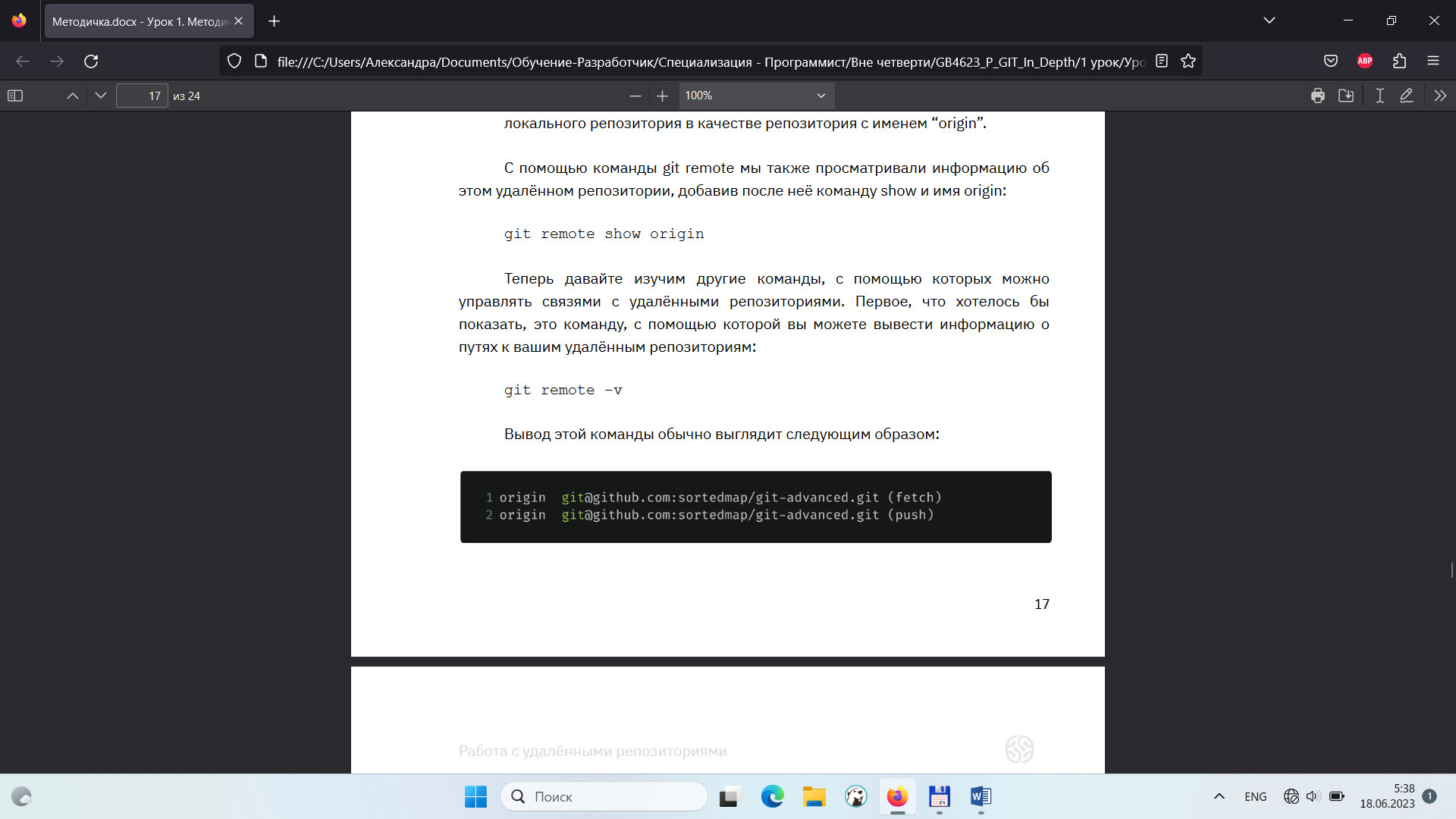
Настройки связей с удалёнными репозиториями

Команды, с помощью которых можно управлять связями с удалёнными репозиториями. Первая команда – это команда, с помощью которой можно вывести информацию о путях к удалённым репозиториям:

git remote –v

В отличие от git remote show origin, она не делает запрос к удаленному репозиторию, а показывает только путь.

Вывод этой команды обычно выглядит следующим образом:



При необходимости вы можете заменить репозиторий origin. Например, поменять путь к этому репозиторию. Для этого сначала нужно удалить текущий репозиторий:

git remote remove origin

И затем снова добавить новый удалённый репозиторий:

git remote add origin path

Вы также можете поменять путь к этому репозиторию, если захотите:

git remote set-url origin new-path

Кроме того, вы можете поменять пути для каждой из операций — fetch и push — отдельно. Например, для операции push команда изменения пути будет такой:

git remote set-url --push origin new-path

Система контроля версий Git также позволяет добавить к своему локальному репозиторию несколько удалённых. Вначале добавим к нашему репозиторию ещё один удалённый:

git remote add vendor vendor-repo-path

После такого действия вы сможете отправлять изменения из своего локального репозитория в какой-то один удалённый репозиторий или сразу в оба. Для отправки изменений в каждый репозиторий отдельно вам нужно просто выполнить последовательно две команды:

git push -u origin master

git push -u vendor master

При этом затянуть изменения из всех репозиториев в свой локальный можно одной командой. Но это можно сделать только без слияния, потому что возможны конфликты не только кода из удалённого репозитория и кодом из локального, но и кодов из двух удалённых репозиториев между собой:

git fetch --all

После такого действия вы сможете просматривать затянутые изменения и сравнивать удалённые ветки с ветками своего локального репозитория:

git log origin/master ^master

git log vendor/master ^master

При необходимости вы можете также вливать нужные вам изменения командной git merge, например:

git merge origin/master

или:

git merge vendor/master

Чтобы однократный “git push” отправлял изменения сразу в несколько репозиториев, нужно придумать название некого виртуального репозитория, который будет в себе содержать пути ко всем удалённым. Пусть он называется “all”. Добавим его тоже с таким же путём, как путь к нашему первому репозиторию:

git remote add all path-1

Дальше вы можете добавить к этому репозиторию пути:

git remote set-url --add --push all path-1

git remote set-url --add --push all path-2

И теперь вы можете отправлять наши изменения сразу во все репозитории, используя имя all:

git push all master

или git push -u all 1-some-issue

Разрешение типичных конфликтов

Конфликты бывают двух видов: простые, которые могут быть разрешены автоматически самим Git-ом (git pull, git push), и сложные, которые можно разрешить только вручную (git pull, ручное редактирование кода).