**Инструменты для работы**

Инструменты, которые могут пригодиться для выполнения заданий на курсах:

1. IDE или текстовый редактор, например:
   1. Visual Studio 2019 Community Edition.
   2. Visual Studio Code с установленным расширением C# for Visual Studio Code (powered by OmniSharp).
2. Система управления версиями git.

Все инструменты рекомендуется инсталлировать с *английским языком* интерфейса.

**AltexSoft LAB git flow**

На протяжении всего курса вы будете выполнять задания, которые будут проверяться на последующих занятиях. Для удобного хранения и ревью выполненных заданий предполагается, что каждый участник заведет **публичный репозиторий на GitHub** и добавит туда лекторов.

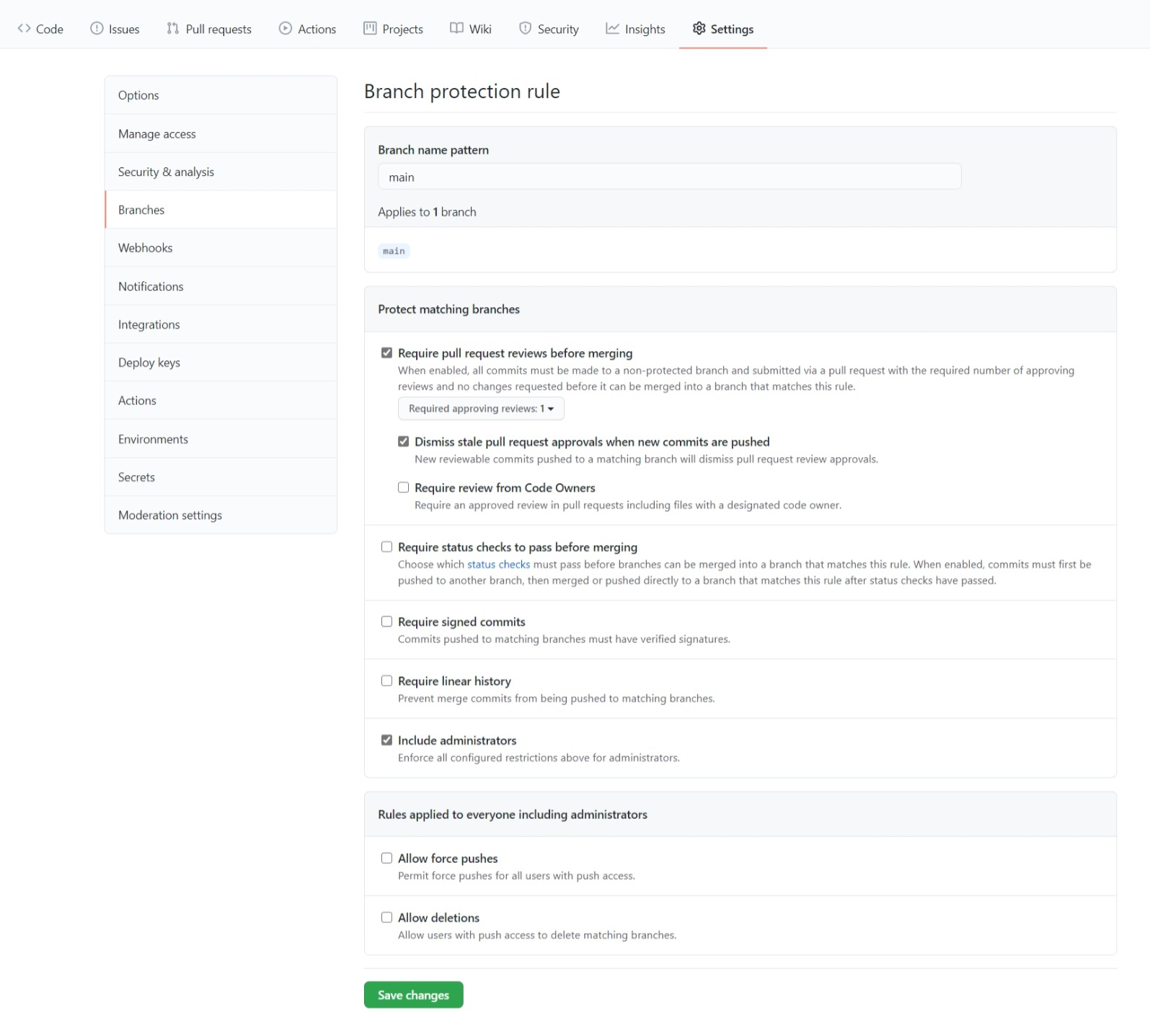
Профили для добавления лекторов:

1. mihail.saharilenko@altexsoft.com
2. yevhen.bilushko@altexsoft.com
3. serhii.shkodenko@altexsoft.com

В репозиторий должен быть добавлен файл *.gitignore с* шаблоном для ***Visual Studio***. Вы можете его добавить прямо во время создания репозитория.

Требования по организации ветвления:

1. Все выполненные и проверенные задания должны находиться в главной ветке, для которой заданы следующие настройки:



Эти настройки запрещают прямые коммиты в ветку *main* для всех, в том числе, администраторов. Так как вы являетесь не только администраторами, но и владельцами репозитория у вас все равно будет возможность пушить изменения напрямую, однако мы вас убедительно просим так не делать.

1. Каждое новое задание следует оформлять в **отдельной ветке**.
2. Каждая ветка должна иметь в качестве базовой главную ветку (если в задании не указано другого).
3. Задания, которые не были оформлены в отдельную ветку, проверяться не будут.
4. Ветка с заданием не может быть вмержена в главную **до** одобрения лектором.
5. **После** одобрения лектором ветка может быть вмержена в главную.
6. **Не будут проверяться** задания, которые были оформлены не с помощью пулл реквеста, а через механизм аплоада файлов.

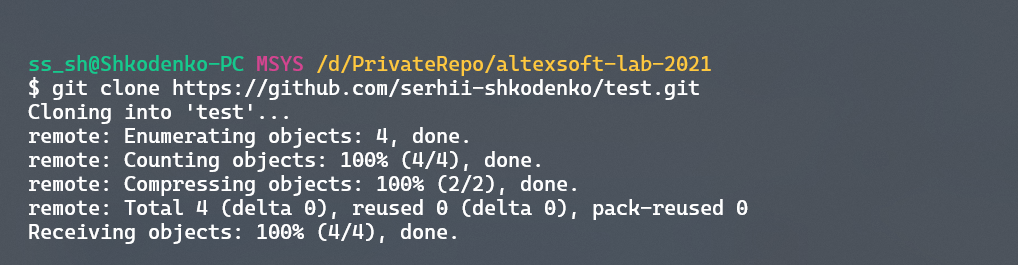
Во время ревью кода лекторы будут оставлять комментарии в коде заданий, оформленных в ветки. Это могут быть комментарии как непосредственно относящиеся к теме задания, так и касательно применения принципов SOLID, C# coding standards, naming conventions, других аспектов.

Чтобы ваш код прошел ревью, вы можете локально в проекте сделать изменения, какие считаете нужным, закомитить их, отправить их в ветку в ремоут репозиторий и отметить комментарий лектора, как исправленный (для этого служит кнопка *Resolve*).

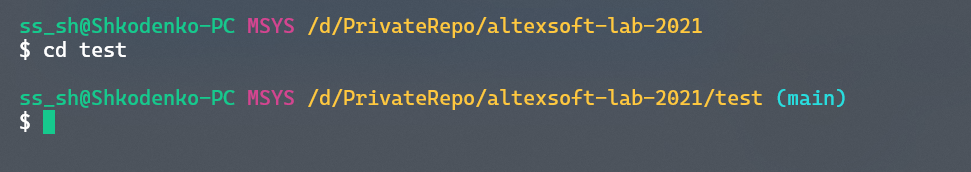
Ниже приводится список комманд git, которые вам могут пригодиться при работе над проектом.

Клонирование **нового репозитория** и переход в папку с загруженными файлами:

git clone [repourl.git]

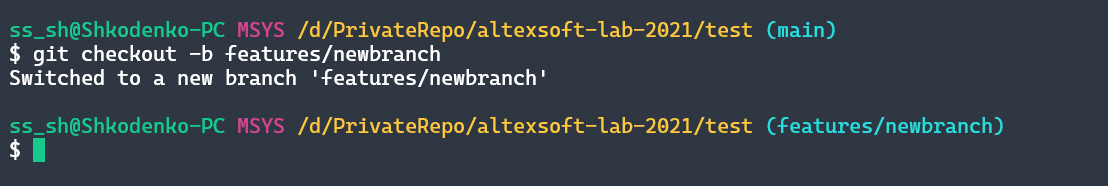


cd [path]



Создание **новой ветки** и быстрое переключение на вновь созданную ветку:

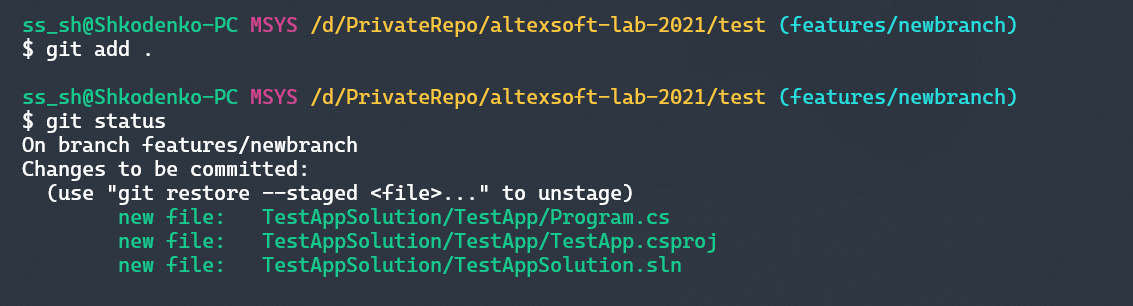
git checkout –b [branchname]



**Добавление файлов** для отслеживания, которые были вам добавлены после начала работы над проектом в текущей ветке и проверка текущего статуса:

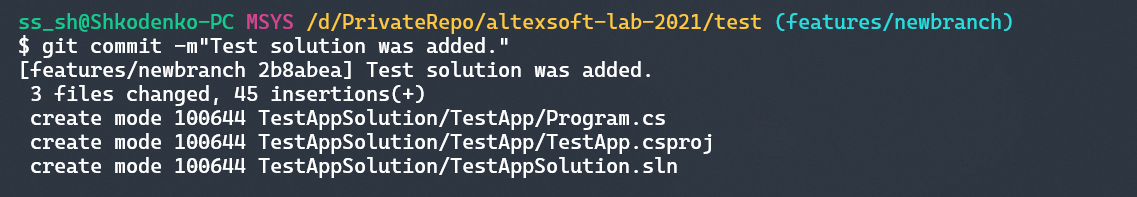
git add .

git status



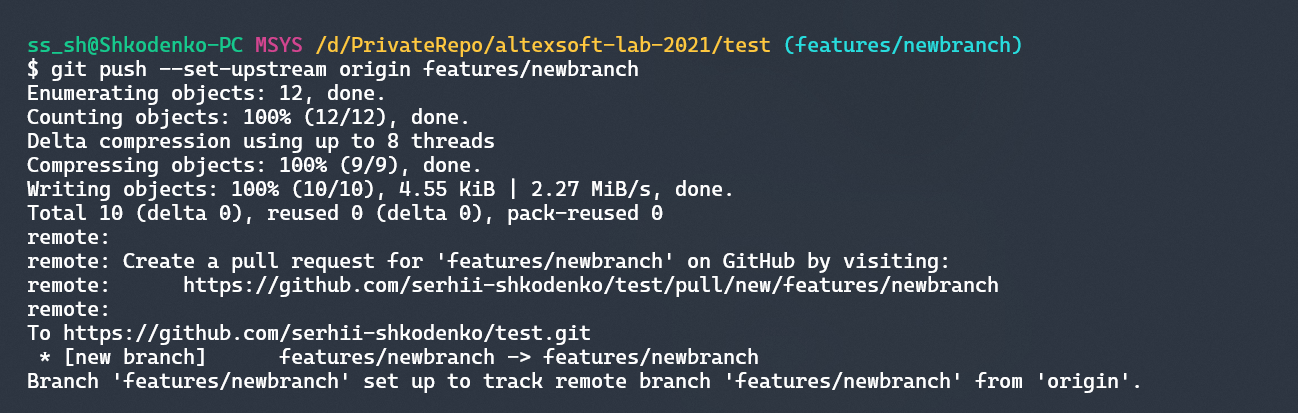
**Коммит** в ветку с указанным сообщением:

git commit –m“[Message].”

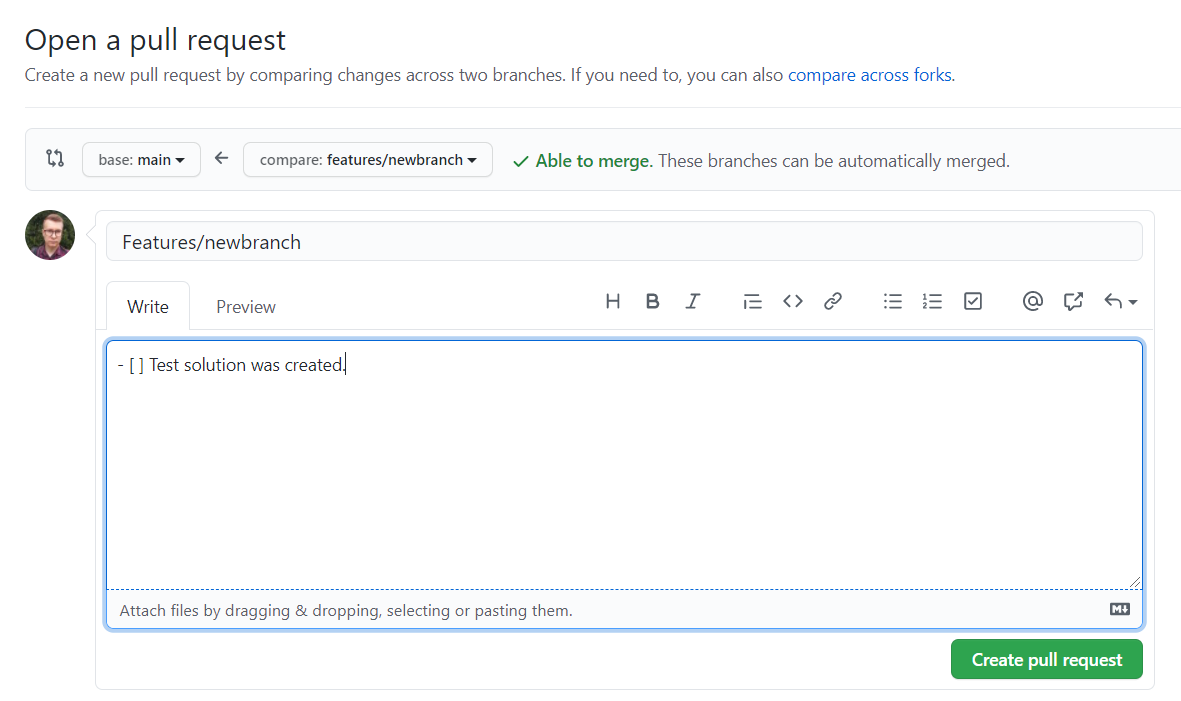


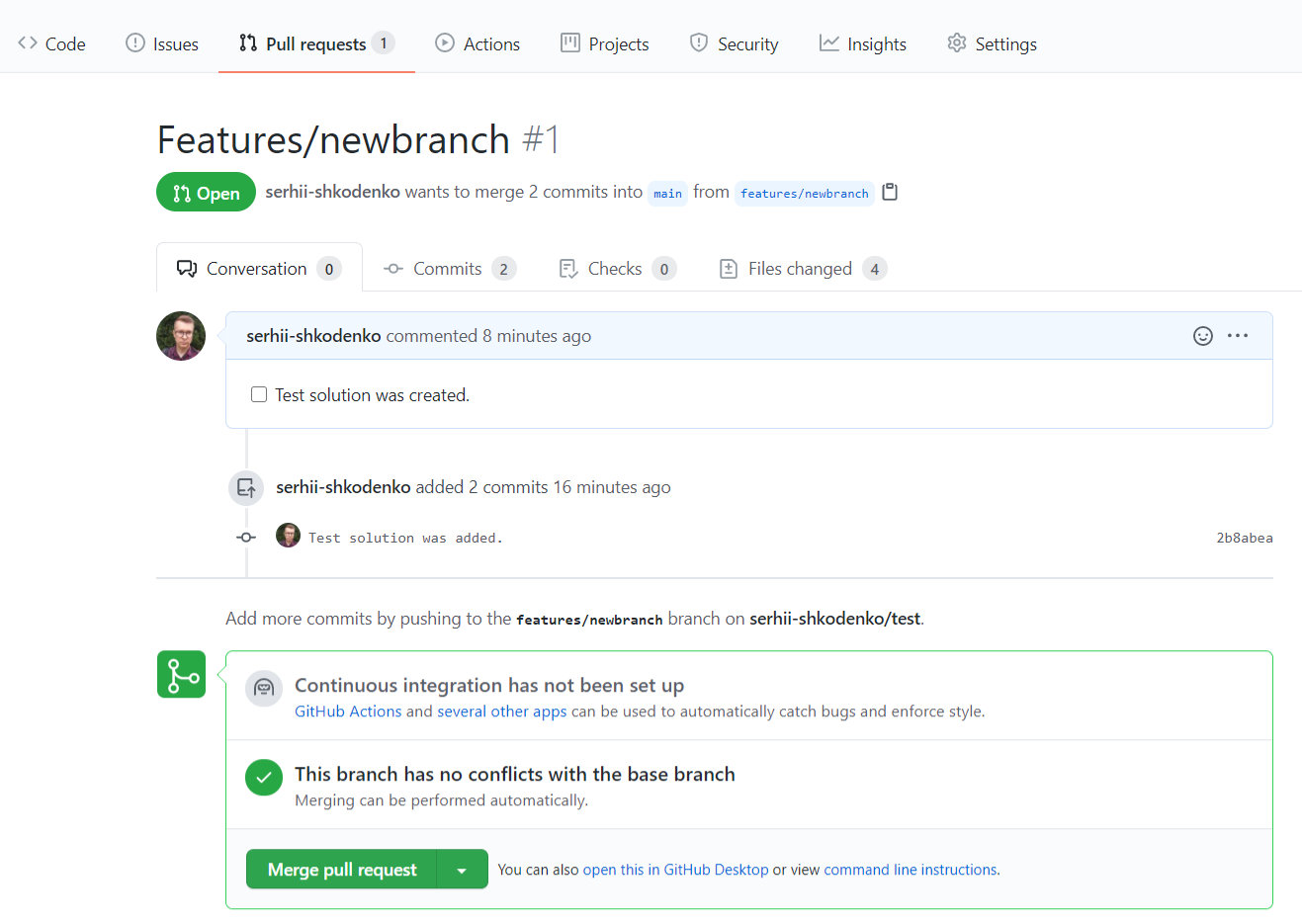
**Отправка** в удаленный (remote) репозиторий осуществленных вами изменений:

git push –set-upstream origin [branchname]



Подготовка в веб-интерфейсе GitHub **пулл реквеста** и ожидание ревью.





После **исправления всех комментариев**, интерфейс пулл реквеста на GitHub будет содержать следующие записи:



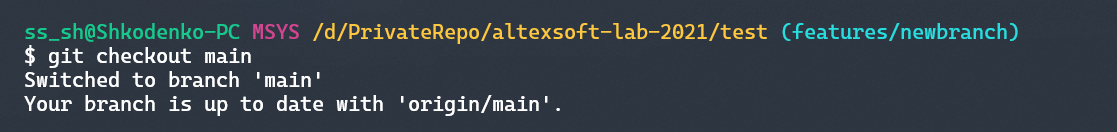
**После исправления всех комментариев**, вы сможете смержить ваш пулл реквест в главную ветку.

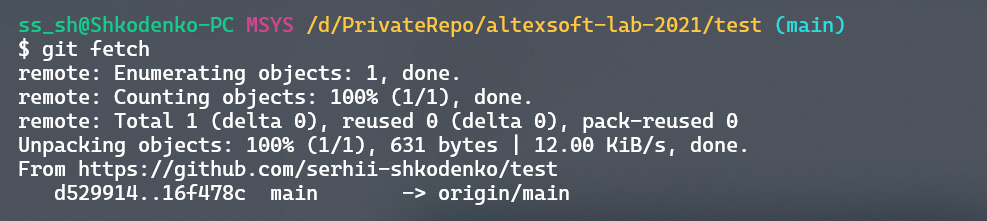
Чтобы изменения в remote репозитории в ветке main **отобразились локально** нужно перейти на эту ветку и запросить изменения:

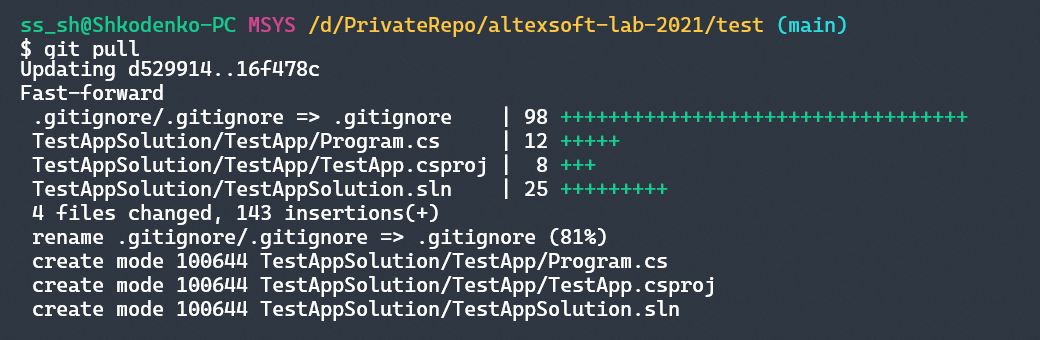
git checkout main

git fetch

git pull







Этот список не является исчерпывающим. В работе, возможно, вам придется использовать другие команды. С ними вы можете ознакомиться в официальной документации <https://git-scm.com/doc>.